UNIVERSIDAD DE GRANADA

E.T.S. DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

Departamento de Ciencias de la Computación
e Inteligencia Artificial

Representación y Tratamiento Semántico de Información Imprecisa en Bases de Datos

Tesis Doctoral

Jesús R. Campaña Gómez

Tutores: J.M. Medina Rodríguez y M.A. Vila Miranda

Granada, Junio de 2011

Editor: Editorial de la Universidad de Granada
Autor: Jesús R. Campaña Gómez
D.L.: GR 614-2012
ISBN: 978-84-694-6676-6

La memoria titulada "Representación y Tratamiento Semántico de Información Imprecisa en Bases de Datos", que presenta D. Jesús Roque Campaña Gómez, para optar al grado de Doctor, ha sido realizada en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección de los Doctores María Amparo Vila Miranda y Juan Miguel Medina Rodríguez.

Granada, Mayo de 2011

El doctorando | Los directores

Jesús R. Campaña Gómez | M. Amparo Vila Miranda | Juan Miguel Medina Rodríguez

Índice general

Capítulo 1

Objetivos e Introducción

En la sociedad actual, la información se está convirtiendo en un activo cada vez mas importante. El uso masivo de Internet ha traído consigo la aparición de nuevas fuentes de datos y ha generado nuevos usos de la información. Está claro que la cantidad de información disponible actualmente es abrumadora, es tal la ingente cantidad de datos, que cada vez parece más complicada su gestión. No cabe duda de que si alguna herramienta ha destacado de manera clara por su capacidad para la gestión de grandes cantidades de datos, esas han sido las bases de datos.

Sin embargo, nos encontramos con que los datos están evolucionando, el disponer de mucha información no representa en absoluto una solución a los problemas de la gestión del conocimiento, siendo más importante obtener el máximo rendimiento de la información que se encuentra disponible. El modelo de gestión de la información está cambiando, experimentando una evolución desde el inicial procesamiento de la información, a un intento por comprender la información. Dentro de esta tendencia existe una corriente de investigación que trata de profundizar en la búsqueda de mecanismos que permitan facilitar la comunicación entre los computadores y las personas.

En los últimos años se están realizando grandes esfuerzos en la gestión de información, que van más allá del mero almacenamiento y recuperación de los datos. La comprensión de la información de forma automática se ha convertido en el objetivo a alcanzar si realmente se quiere llegar a realizar un gestión eficiente de ésta. En este contexto, cada vez más nos encontramos con propuestas que tratan acerca de la gestión de la semántica de los datos. La forma de representación de conocimiento más extendida en ciencias de la computación es la de las ontologías. Aunque el concepto de ontología no es algo nuevo, su aplicación a la gestión de datos en Internet, lo que se conoce como Web Semántica, ha despertado un renovado interés en esta formalización del conocimiento.

En el presente escenario, nos encontramos con herramientas como las bases de datos, que han demostrado ser muy útiles para la representación y gestión de la información tal y como la conocíamos hasta ahora. Aunque también existe un renovado interés por las ontologías y otros mecanismos de representación del conocimiento. La tendencia actual se dirige hacia un cambio en el paradigna de la gestión de contenidos, donde se quiere pasar de metodologías orientadas a los datos, a otras orientadas a la información y su significado.

La propia evolución del concepto de la Web Semántica es un reflejo de este fenómeno. Se está evolucionando desde una web de datos, en donde las bases de datos proporcionan un soporte esencial, a una web de conocimiento donde las ontologías acaparan la atención. Sin embargo, las ontologías como tales siguen siendo datos que han de ser gestionados, almacenados y procesados. Dado que las bases de datos tradicionalmente han sido la solución óptima para la gestión de datos, parece adecuado adaptarlas para tratar con esta nueva forma de gestión de la información basada en ontologías.

En general, el objetivo de este trabajo es profundizar en el estudio del conjunto de técnicas que permitan obtener la semántica de la información en un entorno de base de datos, y que faciliten el acceso y manipulación por parte de un usuario.

Aunque este es el objetivo general que nos hemos propuesto, éste se divide en otros objetivos más concretos que iremos estudiando a lo largo de esta memoria. La pregunta esencial es: ¿A que niveles podemos gestionar la semántica en una base de datos?. En una base de datos tenemos dos componentes claramente diferenciados, por una parte nos encontramos con el esquema de almacenamiento, que determina en que forma se va a almacenar los datos y por otra los datos en sí mismos.

Si reflexionamos un poco acerca del proceso de diseño de una base de datos, podemos ver como las primeras etapas de éste están cargadas de semántica. El primer paso es plantear la definición del problema de gestión de datos que se desea solventar. Tras un análisis detallado, la solución al problema se plantea en términos de un modelo semántico, por lo general el modelo Entidad-Relación, en el cual se plasman las relaciones existentes entre los datos. El model entidad-relación es una descripción de entidades del mundo real. Parte de la semántica de este modelo se pierde al producirse la traslación al esquema lógico relacional. Si se quiere tener información acerca de la semántica del problema se debe consultar el diagrama entidad-relación que ha de formar parte de la documentación del esquema, porque como tal no se almacena en la base de datos. Es más, una vez realizada la transformación el esquema entidad-relación, éste no tiene sentido más que para los diseñadores del sistema. La semántica de la estructura deja de tener relevancia.

Sin embargo, en las ontologías, tanto la propia estructura como los datos son relevantes. Ya que la estructura determina la relación existente entre los datos. En ese aspecto el modelo entidad-relación y la especificación estructural de las ontologías tienen ciertos puntos de conexión. Uno de los objetivos más concretos que nos proponemos, es estudiar los puntos en común existentes y almacenar información sobre la semántica de un esquema de base de datos, dentro de la propia base de datos. Pero, ¿como expresamos esta semántica?, ¿mediante una ontología?. Podemos ir un paso mas allá, si deseamos adaptar un esquema de base de datos para el almacenamiento de una ontología, y esta contiene toda la semántica del problema a modelar, ¿por qué no utilizar la propia ontología para diseñar el esquema de almacenamiento de sus datos?. Y puesto que esta contiene la semántica del problema, si la almacenamos también tendremos el modelo semántico correspondiente al esquema. Evidentemente ésta no es una tarea sencilla, pero puede aportar numerosas ventajas.

Para la **gestión de la semántica a nivel estructural**, proponemos el uso de ontologías como herramientas para el diseño conceptual del esquema de almacenamiento de las instancias de la ontología. Además la propia ontología debe ser almacenada en la base de datos para tener acceso a la semántica de las estructuras. De este modo tendremos va-

rios componentes diferenciados, por una parte el esquema creado a partir de la ontología, donde se almacenan las instancias de la ontología, y por otra parte la propia ontología almacenada. Es necesario aportar información adicional para poder identificar las partes del esquema que se corresponden con las de la ontología. Con toda la información almacenada podríamos realizar tareas básicas de razonamiento como la subsunción de clases.

Aún nos queda realizar el tratamiento de la semántica inherente a las instancias. Por lo general las instancias van a estar compuestas por atributos de tipo numérico y de cadena. El tratamiento semántico de los atributos numéricos lo realizaremos por medio de la consulta. Mediante la aplicación de la lógica difusa a los sistemas de bases de datos es posible cambiar la naturaleza de la interacción entre el usuario y la base de datos. Un usuario puede expresarse de una forma flexible, más parecida a su forma habitual de comunicarse, y la base de datos es capaz de procesarlo. Conceptos numérico como valores aproximados, rangos, o el uso de etiquetas linguísticas, permiten al usuario realizar consultas más ricas.

Por tanto, otro de nuestros objetivos especifícos, es la **gestión de datos numéricos difusos**. La especificación de este tipo de atributos se ha de realizar a nivel de la ontología por lo que habremos de definir nuevos tipos XML Schema.

En cuanto al tratamiento de los campos textuales, encontramos dos tipos diferenciados de atributos aquellos que son identificadores, nombres, direcciones, etc... y texto libre. En el primer caso no dejan de ser identificadores y por tanto no tiene sentido procesar su contenido de forma semántica. Sin embargo el texto libre ofrece interesantes posibilidades. Los campos textuales pueden tener un contenido con semántica propia, imaginemos un campo *Título* en un tabla *Artículo Científico*, el título puede hacer referencia a muy diversos asuntos. Imaginemos que la tabla antes mencionada aparece en una base de datos sobre la que no tenemos información a priori, ¿sobre que tratan estos artículos?. La única forma de responder a esta pregunta es revisando todos los títulos y hacerse una idea de su contenido. ¿Y que ocurriría si queremos obtener artículos sobre *Biología*?. ¿Hacemos una consulta sobre el texto buscando por el término *Biología*?. La solución es realizar el procesamiento de las columnas con atributos textuales y extraer la semántica de éstas, de tal forma que el usuario pueda acceder a dicha información y hacerse una idea sobre el contenido del campo de texto en cuestión.

El siguiente objetivo específico consiste en la **gestión y tratamiento de la semántica en campos textuales**.

Para poder extraer semántica del texto necesitamos procesarlo. No toda la información que aparezca en el texto será relevante por lo que debemos usar criterios para extraer los rasgos comunes a los textos de la columna, en definitiva, realizar una representación del dominio del contenido de la columna. Para poder realizar este resumen utilizaremos técnicas de minería de datos y texto, con el objetivo de crear una representación del texto que pueda ser extendida con semántica obtenida de fuentes externas. Vamos a proponer una metodología general a seguir para el procesamiento semántico de textos. Esta metodología se puede aplicar usando diferentes herramientas.

Una vez aplicada la metodología obtendremos una representación semántica del dominio del problema en forma de ontología. Esta ontología es un resumen del dominio del problema, y además aportará términos de búsqueda para las consultas de usuario por cada uno de los conceptos identificados. Dado que estamos gestionando información relativa a ontologías, la

ontología obtenida se almacenaría dentro del sistema. Sin embargo, deberemos diferenciar la gestión de estas ontologías, puesto que su finalidad y uso difieren respecto a las ontología que definían semántica estructural.

El proceso de extensión de las estructuras de representación de textos a ontologías se puede realizar utilizando diversas fuentes de conocimiento externas. En nuestro caso analizaremos la viabilidad del proceso utilizando WordNet y Wikipedia.

El objetivo específico final consiste en el **estudio de la viabilidad de herramientas como WordNet y Wikipedia para la extensión semántica de textos**.

En resumen, en el presente trabajo estudiaremos la posibilidad de aumentar la semántica de las estructuras de una base de datos y de la información contenida en ésta. El proceso de extensión semántica de la base de datos estará enfocado en dos vertientes:

- *Dotar de semántica a las estructuras de la base de datos*: Partiremos de una representación semántica en forma de ontología de dominio de un conjunto de datos, y lo almacenaremos en la base de datos manteniendo su semántica.

- *Obtener una interpretación semántica de los campos de la base de datos*:

  - *Campos numéricos*: Gestionaremos el almacenamiento y consulta de los campos numéricos empleando lógica difusa. Esto nos permitirá realizar consultas más ricas sobre los datos.

  - *Campos textuales*: Obtendremos información adicional de los campos textuales de la base de datos, aplicando técnicas de minería de datos y obteniendo una representación intermedia, a la que dotaremos de semántica en base a su contenido. Dicha semántica será representada como una ontología, y almacenada en el sistema.

Según el modelo planteado tendríamos acceso a las instancias a través de SQL y la base de datos, mientras que la gestión de la ontología se podría realizar con Protégé. Aún no contemplamos la posibilidad de la evolución de las ontologías por lo que la gestión con Protégé sería externa al sistema, en calidad de generador y visualizador de las ontologías generadas.

Las ontologías con las que vamos a trabajar las representaremos mediante OWL, un estándar de la Web Semántica.

Un resumen del proceso completo puede verse en la Figura 1.1.

El contenido de los sucesivos capítulos está organizado de la siguiente forma:

- Capítulo 2: Realizamos un repaso del estado actual de las diferentes investigaciones en los campos en los que vamos a desarrollar nuestro trabajo. Comenzaremos con el análisis de diferentes trabajos que tratan acerca de la representación semántica de datos en entornos de bases de datos. Analizaremos trabajos sobre almacenamiento de ontologías en bases de datos relacionales y de extracción de contenidos semánticos. Haremos especial hincapié en trabajos que obtienen representaciones semánticas a partir de textos no estructurados.

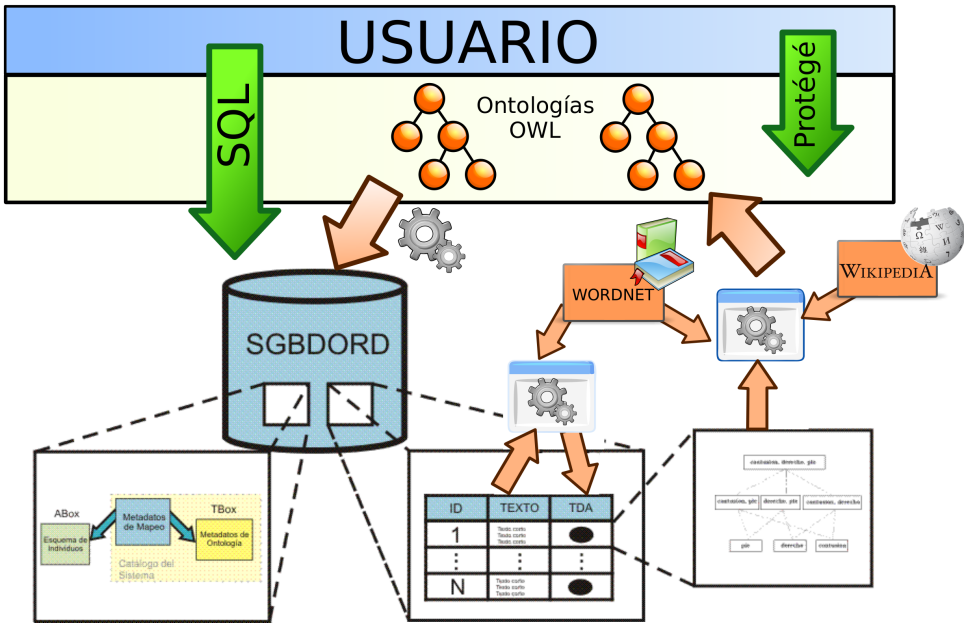


Figura 1.1: Esquema General de la Propuesta

Una vez hemos estudiado las diferentes técnicas y modelos que nos pueden ser de utilidad, los siguientes capítulos describirán algunas de nuestras propuestas.

- *Capítulo 3*: Describe un procedimiento general para transformar ontologías descritas en OWL, en esquemas de bases de datos, manteniendo la semántica. Se presenta una estructura de catálogo para almacenar la ontología, y un algoritmo para transformar la ontología a un esquema.

- *Capítulo 4*: Presentamos una metodología general para extensión semántica de textos no estructurados donde se incluyen etapas de limpieza, representación y extensión semántica de los datos. Además se presenta una implementación de la metodología a modo de ejemplo, utilizando WordNet para la varias etapas del procesamiento incluyendo la extensión semántica.

- *Capítulo 5*: Realizamos una implementación de la metodología usando Wikipedia para la extensión semántica. Además analizaremos diversos métodos para la generación de ontologías de dominio usando Wikipedia.

- *Capítulo 6*: Presentamos los experimentos de evaluación realizados para los métodos de extensión semántica utilizando Wikipedia.

- *Capítulo 7*: Finalmente se presentan las conclusiones y los trabajos futuros a realizar.

Acompañando al documento principal tenemos una serie de apéndices en los que poder desarrollar con mayor extensión aspectos relacionados con el conjunto principal de temas

tratados, pero que con el objetivo de no desviar la atención del núcleo principal del documento, abordamos de forma separada. Los apéndices realizados son los siguientes:

- *Apéndice A*: Realizamos una introducción a todas las materias relacionadas con el núcleo central de la tesis, pero que no han tenido cabida en el discurso general del documento. Estas breves introducciones tienen un componente más divulgativo, puesto que ponen en antecedentes al lector acerca de muchas de las distintas materias que se abordan a lo largo de la memoria. En esta introducción realizamos un repaso a la lógica difusa y a los modelos más importantes de la literatura en cuanto a bases de datos difusas, en sus vertientes relacional y objeto-relacional. También realizamos un breve repaso a las principales características de WordNet.

- *Apéndice B*: Presenta las herramientas utilizadas y desarrolladas durante la confección de esta memoria. De forma breve se describe la función de cada una de las herramientas, así como se dan notas acerca de su uso.

Capítulo 2

Antecedentes

En este capítulo vamos a realizar un breve repaso a trabajos previos realizados en las disciplinas que estamos tratando en esta memoria.

Tal y como hemos visto en el capítulo de introducción, nuestra propuesta combina una gran variedad de técnicas y tecnologías. Trabajaremos con bases de datos y ontologías, y aplicaremos técnicas de aprendizaje ontológico, minería de datos y representación de conocimiento. Por tanto, es conveniente que realizamos un breve repaso del estado del arte de aquellos campos que estén mas relacionadas con el trabajo que vamos a realizar.

Haremos especial hincapié en aquellos trabajos relacionados con el almacenamiento y la extensión semántica de bases de datos. Dado que una de nuestras líneas de trabajo se centra en el almacenamiento se ontologías en bases de datos, revisaremos algunas de las actuales propuestas y formularemos una propuesta adecuada.

Con respecto al componente de generación de ontologías a partir de textos no estructurados, revisaremos el estado de arte actual en este campo y analizaremos algunos trabajos que nos puedan ser de utilidad. Trataremos de dar una visión general de las diferentes líneas de investigación existentes en este campo, así como de las distintas tareas que componen el proceso general de aprendizaje de ontologías a partir de textos.

2.1. Representación Semántica de Datos en BD

La Web Semántica tal y como la concibieron Berners-Lee, Hendler and Lassila en [Berners-Lee et al., 2001] proporciona un entorno de trabajo común en el que los datos son compartidos y reutilizados a través de aplicaciones, comunidades y empresas. El objetivo de la Web Semántica es proporcionar un entorno que permita tanto a humanos como máquinas utilizar sus contenidos de forma eficiente y simple.

Las ontologías son cruciales en el desarrollo de este nuevo modelo de la Web, ya que se presentan como las herramientas clave en el trabajo con contenidos semánticos, gracias a su potencial para describir la semántica de la información y resolver problemas de heterogeneidad.

El uso de forma extendida de ontologías en la Web vino de la mano de la aparición de lenguajes estándar para la representación de ontologías para la Web Semántica. De entre la multitud de lenguajes aparecidos (los cuales se repasan de forma breve en el Apéndice A.4.2) nosotros centraremos nuestra atención en OWL.

OWL (Web Ontology Language) [OWL, 2004] es una recomendación del W3C para el procesamiento de contenidos en la Web. OWL está construido sobre RDF y ambos están escritos en XML. En este trabajo vamos a utilizar definiciones de ontologías expresadas en OWL para aprovechar la amplia variedad de herramientas de gestión disponibles y sobre todo por la condición que ostenta de estándar web. Desde 2009 existe una nueva especificación para el lenguaje OWL conocida como OWL2 [W3C OWL Working Group, 2009, Hitzler et al., 2009]. Esta nueva versión es compatible con versiones anteriores, lo que garantiza que toda ontología expresada en OWL está a su vez expresada en OWL2.

El pleno desarrollo de la Web Semántica se ve restringido por la capacidad disponible para manejar enormes cantidades de datos. Esto supone un problema bastante importante, puesto que los gestores y razonadores actuales no son capaces de manejar eficientemente grandes cantidades de instancias. Las Bases de Datos Relacionales han demostrado ser una de las más eficientes formas de manejar grandes volúmenes de datos, por lo que parece una buena aproximación almacenar las instancias de la ontología en una base de datos.

Si los datos sobre instancias se van a gestionar en la base de datos, será necesario definir un esquema para almacenarlas. Existen dos aproximaciones diferentes a la hora de abordar el problema, almacenar la ontología completa en tablas específicamente diseñadas para ello o crear un esquema de bases de datos basado en la definición de la ontología para almacenar las instancias. Cada una de estas aproximaciones tiene sus ventajas e inconvenientes como se explicará más adelante. Con el objetivo de compensar las desventajas de cada aproximación, nuestra propuesta usa una combinación de ambas aproximaciones para desarrollar un esquema combinado para almacenar la ontologías y sus instancias sin pérdida de información.

2.1.1. Ontologías como herramientas para el diseño de Bases de Datos Relacionales

Las Bases de datos Relacionales van a formar una parte importante de la Web Semántica tal y como ya lo forman de la Web actual. Probablemente la mayor parte de la información

de la Web Semántica terminará siendo almacenada en bases de datos relacionales. La mayor parte de de la información actualmente presente en las bases de datos relacionales formará parte de la Web Semántica a través de mapeos de ontologías y transformaciones, pero también se desarrollarán nuevas aplicaciones desde cero.

Los datos de la Web Semántica dependen fuertemente de definiciones ontológicas. De hecho, los datos se componen de instancias de ontologías. Por lo tanto, cuando se desarrolla una nueva aplicación y es necesario diseñar un esquema de base de datos para almacenar las instancias, una ontología de dominio describiendo los datos de la aplicación puede usarse como modelo conceptual para el diseño de la base de datos relacional. Una de las principales diferencias entre las ontologías y las bases de datos relacionales, es que las primeras hacen una suposición de mundo abierto (una afirmación no conocida puede ser cierta) mientras que las segundas utilizan una semántica de mundo cerrado (toda afirmación no conocida es falsa). Esta característica no es particularmente relevante en este caso, pero será tenida en cuenta en caso de realizar futuras tareas de razonamiento.

La relación entre esquemas de bases de datos existentes y ontologías a través de mapeos, ha sido ampliamente tratada en la literatura [Barrasa et al., 2004, Bizer and Seaborne, 2004]. Pero no existe tanta profusión de trabajos empleando ontologías como modelos conceptuales.

El trabajo [Trinkunas and Vasilecas, 2007] presenta una transformación automática de ontología a un modelo conceptual Entidad-Relación. El acercamiento basado en grafos, convierte una ontología expresada en OWL DL en un modelo ER. La generación del DDL y DML se delegan en una herramienta comercial. El problema con esta aproximación es que una vez se ha diseñado el sistema la ontología deja de tener utilidad, puesto que no hay forma de establecer una correspondencia entre el esquema generado y la ontología original. Debido a este inconveniente, el método parece poco apropiado para el diseño de sistemas para la Web Semántica.

En [Jean et al., 2006] se realiza un profundo e intuitivo repaso de las ontologías de dominio desde una perspectiva de bases de datos. Se propone una definición más completa del término ontología de dominio y se presenta una taxonomía de este tipo de ontologías. También se debate acerca del uso de ontologías como modelos conceptuales. Una ontología de dominio puede usarse como un primer nivel de la especificación de los conceptos de una base de datos, que posteriormente puede refinarse para incorporar requisitos particulares y por tanto definir un modelo conceptual.

Existen múltiples trabajos en los que se presenta la idea de las ontologías como modelos conceptuales. En [Roldán García et al., 2005] se presenta una metodología de diseño para sistemas de bases de datos para la Web Semántica. Esta metodología es empleada para diseñar nuevas bases de datos y para añadir semántica a sistemas existentes. De entre las diferentes ideas presentadas, nosotros nos centraremos en aquella que usa una ontología como punto inicial en el proceso de diseño de un esquema de bases de datos. Se presenta una ontología de dominio como una abstracción del conocimiento de los esquemas de la

fuente de datos. Tras un proceso de refinamiento la ontología se adapta a un esquema local. El proceso de refinamiento comprende dos pasos. En el primer paso, el diseñador extiende la ontología inicial proporcionando nuevos conceptos, relaciones y restricciones de integridad relevantes para el sistema desarrollado. A continuación, un subconjunto de la ontología obtenida en el paso previo se selecciona y se renombra en orden. La nueva ontología obtenida será el punto de partida para implementación del esquema de la base de datos.

En el trabajo [Sugumaran and Storey, 2006] se estudia en profundidad el papel de las ontologías de dominio en el diseño de bases de datos. En este caso la aproximación es ligeramente diferente, la ontología se usa para aconsejar al diseñador acerca de la corrección y completitud del modelo conceptual que ha creado utilizando el conocimiento del dominio contenido en la ontología.

Teniendo en cuenta todos estos trabajos previos, proponemos el uso de ontologías OWL DL como formalismo para la definición de modelos conceptuales para el diseño de bases de datos. De este modo, el diseño de un esquema de base de datos se ve reducido al correcto diseño de la ontología de dominio que capture todos los requisitos de los datos. El diseño de la ontología puede realizarse utilizando herramientas CASE tales como Protégé [Pro, 2008] y SWOOP [Kalyanpur et al., 2006]. Una vez definida la ontología, a través de un proceso de transformación, la ontología se expresa como un conjunto de sentencias DDL y DML que implementan el esquema de una base de datos relacional.

Recientemente en [Al-Jadir et al., 2010] se ha propuesto el sistema OntoMinD una extensión del SGBD Oracle para el almacenamiento de ontologías de gran tamaño, que proporciona capacidades de razonamiento mediante el uso de procedimientos PL/SQL almacenados. Estas tareas de inferencia incluyen subsunción de clases y de propiedades, satisfacibilidad de clases y consistencia de la ABox. Las tareas de razonamiento se realizan en tiempo de actualización y materializan todo el conocimiento inferido dentro de la propia base de datos. Este trabajo opta por evitar la inferencia en tiempo de consulta para agilizar las consultas, ya que es la tarea más frecuentemente realizada sobre ontologías y bases de datos. OntoMinD ofrece soporte para ontología DL-Lite(R,$\sqcap$). Este tipo de ontologías son más restrictivas que las OWL-Lite ya que no soportan cualificación universal y únicamente soportan cuantificación existencial sin cualificar ($\exists P$ en lugar de $\exists P.C$).

El almacenamiento de la ontología se realiza en un metaesquema en la base de datos. Las instancias se almacenan en las tablas creadas para cada una de las clases de la ontología. Estas tablas de clases contienen cuatro atributos *C(fact_id, indiv_id, asserted, inferred)* para almacenar los individuos e indicar si han sido obtenidos por un proceso de inferencia. Por cada propiedad se crea también una tabla *P(fact_id, domain, range, asserted, inferred)* donde se almacenan las instancias de las propiedades, indicando su dominio y rango entre otras propiedades. También se dispone de una tabla adicional *A_Resource(resource_id, name)* que enlaza identificadores de individuos, con los nombres de los individuos.

2.1.2. Transformación de Ontología a Base de Datos

En este apartado vamos a repasar las aproximaciones más importantes empleadas en el almacenamiento de ontologías en bases de datos relacionales. Los métodos que vamos a ver en este apartado tratan más con la parte algorítmica del proceso de transformación de una ontología, mientras que en el caso del apartado anterior se trataba más con la idea del diseño conceptual basado en ontologías.

En [Das et al., 2004] se propone un método para dar soporte de concordancia semántica a un SGBD. En el artículo se propone un método para almacenar una ontología OWL DL en tablas definidas por el usuario en una base de datos relacional y un conjunto de operadores *SQL* para realizar las operaciones de concordancia semántica. Además se presenta un novedoso esquema de indexación para mejorar la velocidad de la nuevas operaciones propuestas. Esta aproximación almacena la ontología junto a sus instancias, todas juntas en tablas del sistema definidas por el usuario, sin que haya pérdida de información.

Siguiendo con la idea de la gestión de instancias de ontologías empleando bases de datos, en [Vysniauskas and Nemuraite, 2006] se presenta una representación de una ontología de dominio OWL en forma de base de datos relacional. Las clases OWL y las propiedades se transforman en tablas y relaciones de un esquema en el que posteriormente se insertarán las instancias. Se crean algunas tablas adicionales para mantener metadatos relacionados con las restricciones impuestas. Esta aproximación es muy interesante, pero adolece de una adecuada representación de la propia ontología, lo cual limita incluso las capacidades más básicas de razonamiento sobre los datos, ya que alguna información es difícil de restaurar tras la transformación.

Mientras esta última aproximación trata de aprovechar los beneficios que ofrece un SGBD en cuanto almacenamiento, pero pierde la capacidad de razonar sobre los datos, la primera propuesta trata adecuadamente el problema del almacenamiento de la ontología pero falla a la hora de organizar un acceso eficiente a los datos.

En [Astrova et al., 2007] se presenta un procedimiento similar al del caso anterior sólo que en esta ocasión está expresado en forma de reglas a aplicar sobre ontologías OWL para convertirlas en esquemas de base de datos y muestra una tabla con las equivalencias entre tipos XSD y tipos SQL. Esta aproximación únicamente se encarga de transformar la ontología en un esquema de base de datos válido, una vez conseguido esto se prescinde de la ontología original. Además no se tienen en cuenta casos en los que nos encontremos con clases complejas.

De forma similar al trabajo anterior en [Bagui, 2009] se presentan una serie de reglas para realizar la transformación desde una ontología OWL a un esquema Entidad-Relación extendido. En este caso en lugar de generar el esquema directamente, se genera un modelo semántico a partir de la ontología. Dado que se trata únicamente de generar un modelo de semántico, no se especifica en ningún momento el tratamiento que se ha de realizar con las posibles instancias que incluya la ontología.

2.1.3. Extracción de Ontologías en Bases de Datos Relacionales

Mientras que la cantidad de trabajos centrados en la transformación de ontologías en esquemas de bases de datos es relativamente escasa, no ocurre lo mismo para el caso inverso. Existe una gran cantidad de trabajos y herramientas que tratan la necesidad de expresar un base de datos existente en términos de una ontología. Aunque a lo largo de este trabajo trataremos con el diseño de esquemas a partir de ontologías de dominio, es muy interesante ver las distintas aproximaciones que tratan con el problema inverso, de modo que podamos trasladar algunas de las soluciones empleadas a nuestro problema particular.

D2R MAP [Bizer, 2003] es un lenguaje declarativo basado en XML, que describe el mapeo entre un esquema de base de datos y una ontología OWL. Partiendo de un esquema de base de datos y de una ontología existentes, se crea un documento que contiene reglas para mapear la base de datos en la ontología. Las reglas de mapeo pueden contener consultas SQL, lo que permite reflejar relaciones complejas en los mapeos (esto nos permite pensar que en el proceso inverso, podríamos expresar relaciones complejas de la ontología como consultas o vistas sobre los datos).

El desarrollo del documento con las reglas de mapeo ha de hacerse acorde con un esquema predefinido, este documento es un documento XML y por tanto su contenido ha de regirse por la estructura determinada en el DTD o XSD correspondiente. El desarrollo del documento de mapeo debe realizarse por parte de un experto, pero una vez diseñado el conjunto de reglas estas son analizadas por un procesador D2R que se encarga de construir la ontología de forma automática a partir de ellas.

D2RQ [Bizer and Seaborne, 2004] es un lenguaje declarativo para describir mapeos entre esquemas de bases de datos y ontologías RDF/OWL. D2RQ permite a las aplicaciones tratar bases de datos no RDF como grafos virtuales que pueden consultarse utilizando RDQL. D2RQ está construido teniendo en cuenta las experiencias obtenidas con D2R y dando un paso más allá, permitiendo no solo la representación de los datos, si no también su consulta desde el punto de vista de la ontología.

Partiendo de un esquema de base de datos y una ontología existentes, se crea un mapeo D2RQ que establece la correspondencia entre ambas estructuras. D2RQ se implementa como un grafo en Jena. Un grafo D2RQ une una o mas bases de datos relacionales en un grafo virtual de solo lectura. Cuando se realiza una consulta RDQL a la ontología, esta es reescrita en forma de SQL teniendo en cuenta las reglas de mapeo. Los resultados obtenidos por la consulta se transforman en triples RDF.

D2R Server [Bizer and Cyganiak, 2006] es una herramienta que emplea el lenguaje D2RQ para publicar el contenido de bases de datos relacionales en la Web Semántica. D2R Server permite a los navegadores HTML y RDF, navegar por el contenido de bases de datos no RDF, permitiendo a las aplicaciones consultar la base de datos utilizando el lenguaje SPARQL.

El principal aporte respecto a los anteriores trabajos es la automatización de la generación de los mapeos y la posibilidad de realizar consultas en SPARQL.

En cuanto a la generación automática de mapeos, se establece el mapeo de las diferentes tablas en clases, y de las columnas en propiedades. Posteriormente existe la posibilidad de sustituir los términos generados de forma automática por términos más conocidos extraídos de vocabularios RDF predefinidos.

En [Stojanovic et al., 2002a, Stojanovic et al., 2002b] se presenta un procedimiento semi-automático para el mapeo de un esquema de bases de datos a una ontología existente, realizando un proceso de ingeniería inversa. El objetivo es compartir en la Web Semántica, los datos almacenados en la base de datos. Se define una arquitectura para proporcionar una traslación de datos relacionales a instancias RDF de una ontología, bajo demanda.

Se proporcionan la descripción del proceso de mapeo y el conjunto de reglas que van a permitir la transformación del esquema en una ontología. El proceso de aplicación de las reglas de transformación es semi-automático, ya que pueden producirse circunstancias en las que es posible aplicar mas de una regla, en estas ocasiones es cuando se necesita la intervención de un experto para solventar las ambigüedades que surjan.

Existen diferentes conjuntos de reglas, para la formación de conceptos, para la herencia de conceptos y para la formación de relaciones ontológicas.

- **Reglas para conceptos**
  De forma general cada relación se transforma en un concepto en la ontología. Excepto en los casos C1 y C2.
  - C1 - Identifica las relaciones muchos a muchos (n:m).
  - C2 - Integra información sobre una entidad que se encuentran diseminada en varias tablas, y representa un único concepto.
  - C3 - Es la regla por defecto, si las dos anteriores no se pueden aplicar se aplica esta.
- **Reglas para herencia**
  - I1 - Esta regla crea relaciones de herencia si existe una dependencia de inclusión entre dos relaciones y hay definidos conceptos para ambas relaciones. Esta regla es una especialización de C2. El usuario debe decidir entre aplicar C2 o I1.
- **Reglas para relaciones**
  - A1 - Se corresponde con el caso C1. Se crean dos propiedades inversas entre sí, que enlazan los conceptos creados a partir de las relaciones que unían en el esquema de base de datos.
  - A2 - Se refiere a relaciones uno a muchos (1:n), se generan dos propiedades una simple-valuada en el concepto correspondiente a la parte 'uno', y otra multivaluada en el otro concepto.
  - A3 - Se corresponde con C2 y grupos de atributos que se distribuyen en varias relaciones.

- A4 - Trata relaciones uno a uno (1:1) definiendo una relación simple-valuada entre los conceptos.
- A5 - Relaciones no binarias se descomponen en conjuntos de relaciones binarias tratables.
- A6 - Se aplica en caso de que ninguna de las anteriores reglas pueda aplicarse. Transforma los atributos en relaciones en la ontología.

Una vez definido el mapeo entre el esquema y la ontología, las tuplas son transformadas en instancias de la ontología.

La definición de una arquitectura de transformación y la implementación de los prototipos aquí propuestos, se pueden ver en [Volz et al., 2004].

KAON (Karlsruhe Ontology and Semantic Web Tool Suite) [Bozsak et al., 2002] es una herramienta que proporciona una infraestructura para el diseño, creación, uso y acceso de aplicaciones semánticas. KAON dispone de dos conjuntos diferenciados de herramientas, uno que puede ser utilizado directamente por los usuarios y otro que da servicio a las aplicaciones.

Las herramientas disponibles para los usuarios son:

- KAON Portal : Generador de portales Web basados en ontologías. Dada una ontología, se genera una aplicación web de forma automática.
- OntoMat Application Framework : Es una plataforma que incluye los siguientes componentes
  - OntoMat-SOEP - Entorno multilingüe para la ingeniería y evolución de ontologías, para el desarrollo y mantenimiento manual de ontologías.
  - OntoMat-REVERSE - Ingeniería inversa de esquemas de bases de datos a ontologías, utiliza el algoritmo descrito en [Stojanovic et al., 2002b].
  - OntoMat-SILVA - Implementa una metodología exhaustiva para mediación y mapeo de ontologías consistente en normalización de ontologías heterogéneas, detección de similitudes, representación gráfica de las asociaciones y enlaces semánticos entre dos ontologías.
  - OntoMat-Catyrpel - Herramienta de soporte al usuario para el descubrimiento de metadatos basados en RDF en la Web.

Los servicios disponibles para las aplicaciones incluyen:

- KAON API - Librería para la gestión de ontologías, proporciona acceso y persistencia en sistemas de almacenamiento externos, tales como ficheros o bases de datos relacionales.
- KAON Server - Proporciona repositorio RDF persistente, transaccional y seguro, accesible por varios usuarios de forma simultanea.

En [Volz et al., 2004] se presenta un entorno de generación de metadatos y una arquitectura para la generación de páginas web a partir de una base de datos. Se realiza la anotación del esquema de la base de datos, o bien se anota la representación web generada a partir de la base de datos. Esta aproximación se basa en el hecho de que la mayoría de contenido web se genera de forma automática a partir de datos almacenados en una base de datos. Se describe el proceso de "anotación profunda" como un proceso mediante el cual se accede a la web profunda (*Deep Web*) con el fin de hacer accesible la información desde la Web Semántica, combinando las posibilidades de la anotación tradicional y las bases de datos.

El proceso de anotación profunda, se resume en cuatro etapas:

1. Partiendo de un sitio Web generado a partir de una base de datos relacional, se producen páginas Web anotadas en el servidor teniendo en cuenta la estructura de la base de datos.

2. Un anotador produce anotaciones del lado del cliente, ajustándose a una ontología descrita por el cliente, bien a través de la anotación de las páginas Web o realizando un mapeo automático del esquema de la base de datos a la ontología. De esta etapa se obtiene un conjunto de reglas de mapeo entre la base de datos y la ontología del cliente.

3. El anotador publica la ontología del cliente y las reglas de mapeo derivadas de las anotaciones, de este modo se hacen públicas en la Web.

4. El anotador evalúa y refina el mapeo empleando ciertas directrices. Una vez terminado el proceso las nuevas reglas se publican en la web.

El proceso de anotado se realiza utilizando una herramienta llamada OntoMat-Annotizer en el caso de la anotación de las páginas, y de la herramienta OntoMat-Reverse para la traslación semi-automática del esquema de bases de datos a la ontología.

El proceso de generación de reglas entre la anotación de las páginas y la ontología del usuario, se realiza de forma manual por parte del usuario empleando una herramienta de anotación (OntoMat) que muestras las marcas de las páginas y permite asociarlas a elementos en la ontología.

En el caso en que el mapeo se realice desde el esquema, éste se hace de forma automática por la herramienta OntoMat-Reverse, y luego puede ser modificado por el usuario. El proceso automático de transformación de esquema de base de datos a ontología es realizado según lo descrito en [Maedche et al., 2002] y [Stojanovic et al., 2002b].

El entorno propuesto también permite la resolución de consultas.

En [Handschuh et al., 2001, Handschuh and Staab, 2002] se describe el entorno CREAM (Creating RElational, Annotation-based Meta-data) el cual proporciona utilidades para la anotación manual de de páginas web con metadatos relacionales. OntoMat es la implementación de CREAM. S-CREAM [Handschuh et al., 2002] es una extensión del entorno CREAM que proporciona anotación semiautomática de páginas web. S-CREAM incluye el

componente de extracción de información Amilcare, el cual aprende reglas de extracción de información a partir de un conjunto de entrenamiento compuesto por documentos anotados manualmente.

En [Barrasa et al., 2003] se aborda el problema de publicar los contenidos de una base de datos siguiendo los criterios de la Web Semántica. Se ha desarrollado una aplicación que, a partir de una base de datos de subvenciones y una ontología que describe el dominio tratado, transforma los datos almacenados y los publica de acuerdo a las directrices de la Web Semántica. La transformación de esquema relacional a ontología se describe mediante el uso de un lenguaje declarativo denominado eD2R (extended D2R). De forma adicional se presenta la arquitectura de aplicación, y las principales características del lenguaje de mapeo. El punto de partida supone la existencia de un esquema de base de datos y una ontología predefinidos. A través del lenguaje eD2R se realiza el mapeo entre las estructuras del esquema y la ontología. El proceso de mapeo es realizado de forma manual.

R2O [Barrasa et al., 2004] es un lenguaje extensible y declarativo que permite describir mapeos entre esquemas de bases de datos relacionales y ontologías. Partiendo de una base de datos existente y de una ontología ya definida, se generan una serie de mapeos entre base de datos y ontología, descritos mediante el lenguaje R2O. El objetivo es hacer generar la información almacenada en la base de datos como contenido de la Web Semántica.

R2O permite expresar mapeos que van desde los más simples que identifican una tabla con un concepto de la ontología, a otros más complejos que asocian una consulta sobre el esquema con un concepto de la ontología. Esencialmente se distinguen tres casos en los que los componentes del esquema de base de datos se pueden mapear con la ontología.

- **Caso 1:** Una tabla de la base de datos se mapea a un concepto de la ontología. Cada uno de los registros de la tabla se corresponderá con una instancia del concepto. Con los datos de la tupla se deben rellenar los valores de los atributos de la instancia de la ontología.
- **Caso 2:** Una tabla de la base de datos se usa para instanciar más de un concepto en la ontología, pero solo se rellenará una instancia por cada concepto.
- **Caso 3:** Una tabla de la base de datos se usa para instanciar más de un concepto en la ontología, y se pueden generar múltiples instancias de la ontología. Por cada registro de la base de datos se pueden generar varias instancias de cada concepto.

En algunos casos no todos los atributos de una tabla son necesarios para la generación de los conceptos, de igual modo puede que no todos los registros de una tabla sean necesarios para generar las instancias del correspondiente concepto. En ambos casos antes de generar las instancias de la ontología es posible realizar algunas operaciones algebraicas relacionales. Las siguientes operaciones pueden ser utilizadas:

- **Mapeo directo**. La tabla se mapea en un concepto de la ontología y cada registro se corresponde con una instancia de dicho concepto.

- **Reunion/Join**. Un conjunto de tablas se corresponden con un concepto tras ser reunidas. Todos los registros de la tabla reunida se corresponden con instancias del concepto.
- **Proyección**. Un subconjunto de los atributos de una tabla son los que se corresponden con el concepto de la ontología.
- **Selección**. Un conjunto de tuplas se corresponden con un concepto.
- Cualquier combinación de las operaciones anteriores se corresponde con una operación válida.

El artículo presenta un resumen detallado de las estructuras que componen el lenguaje R2O, así como un apéndice con la su notación BNF. Dado un sistema de bases de datos relacional se describe un algoritmo formal para transformar los metadatos y las restricciones estructurales en una definición de ontología OWL. La ontología generada se describe de acuerdo a un conjunto de vocabularios empleados para definir sistemas de bases de datos relacionales en forma de ontología.

RDB2ONT [Trinh et al., 2006] es una herramienta que genera y publica de forma dinámica ontologías a partir de los metadatos y restricciones estructurales de una base de datos, preservando las restricciones estructurales. La idea básica consiste en publicar el esquema de la base de datos en OWL-RDBO, una ontología para la descripción de bases de datos. Esta ontología contiene vocabularios para describir sistemas de bases de datos en la red, relaciones semánticas entre vocabularios y restricciones en los vocabularios y sus relaciones semánticas.

Esta propuesta difiere del resto en cuanto se trata más de una forma de representar la estructura de la base de datos, que de poblar una ontología con los datos contenidos en la propia base de datos. El trabajo asume que los mapeos entre la ontología generada y otras ontologías son definidos por expertos de forma manual o semi-automática.

En [Konstantinou et al., 2006] se estudia una metodología para el mapeo de contenidos de bases de datos relacionales a ontologías. Se ha desarrollado una herramienta gráfica para asistir el procedimiento de mapeo y exportación de las ontologías enlazadas con la base de datos. Adicionalmente es posible realizar consultas realizadas con lenguajes semánticos a la base de datos, a través de la conexión establecida entre la ontología y el esquema de base de datos.

El método de mapeo consiste en cuatro etapas.

1. Se obtienen un conjunto de datos de la base de datos, a través de una consulta SQL.
2. Se selecciona una clase de la ontología.
3. Se comprueba la consistencia, se evalúa, valida y refina el mapeo. Esta etapa se realiza una vez se han definido todos los mapeos, y realiza pruebas de consistencia tales como comprobar que clases disjuntas no contienen tuplas de la base de datos en común.

4. Se modifica la ontología y se añade información adicional. Como propiedad de una clase se define una cadena que contiene la consulta mediante la cual se recuperarán de la BD las instancias correspondientes a la clase.

Las reglas de mapeo deben ser definidas por un experto con conocimiento del dominio tratado.

La implementación del sistema se ha realizado como un plug-in del editor de ontologías Protégé. Las bases de datos soportadas por el sistema son PostgreSQL y MySQL, a través de JDBC.

RDB2Onto [Laclavík, 2006] es una herramienta para la generación de metadatos semánticos a partir de una base de datos relacional. Dichos metadatos se expresan en forma de una ontología RDF/OWL. La propuesta se centra casi exclusivamente en la recuperación de los individuos para la ontología. A través de una consulta SQL se recuperan de la base de datos un conjunto de tuplas que serán estructuradas en forma de individuos de una ontología. Esta estructuración se realiza rellenando con los resultados obtenidos por la consulta, los huecos de un documento RDF/OWL previamente definido. Se presenta como una alternativa simple a lenguajes de mapeo tales como DR2MAP y R2O.

Aunque esta aproximación permite extraer los datos de la base de datos relacional, y expresarlos como una ontología, carece de la gran mayoría de características deseables en este tipo de sistemas. El principal inconveniente consiste en la imposibilidad de automatizar la tarea de extracción de datos, ya que tanto las consultas SQL empleadas, como la plantilla de la ontología creada, deben ser desarrolladas por un experto que tenga conocimiento del esquema de la base de datos. Del mismo modo, la estructura de la ontología ha de ser desarrollada por el experto. La falta de automatización del proceso hace que esta aproximación no aporte resultados relevantes, pudiéndose utilizar únicamente como una solución ad hoc para casos muy concretos.

En [An et al., 2005] se describen los principios subyacentes, los algoritmos, y un prototipo de una herramienta que infiere mapeos semánticos dadas correspondencias simples entre las columnas de una tabla en un esquema de base de datos y propiedades de tipo de datos en una ontología. Todo el proceso se realiza teniendo en cuenta información del esquema de base de datos (claves primarias, claves exernas, ...) y la ontología (restricciones de cardinalidad, jerarquías is-a). Se desarrolla una herramienta que asiste al usuario en la tarea de definir mapeos entre un esquema de base de datos y una ontología. El algoritmo infiere la semántica implícita en la traducción de un esquema entidad-relación (ER) a un diseño relacional, si el esquema sigue los principios de diseño de ER.

La diferencia de esta aproximación con respecto a la ingeniería inversa, es que aquí se parte de una ontología existente y se quiere interpretar un esquema relacional en términos de ella, mientras la ingeniería inversa trata de construir una nueva ontología a partir del esquema existente.

Se infieren fórmulas complejas expresando el mapeo semántico definido como correspondencias simples entre la base de datos y la ontología.

El algoritmo de inferencia de mapeo se ha implementado en un prototipo de sistema llamado MAPONTO.

En [An et al., 2006], los usuarios proporcionan correspondencias simples entre elementos atómicos de la base de datos (nombres de columnas de una tabla) y los de la ontología (atributos, propiedades de datos, nombres de conceptos). Dado un conjunto de correspondencias MAPONTO realiza un proceso de razonamiento sobre el esquema de la base de datos y la ontología, como salida se genera una lista de reglas candidatas para cada componente individual (tabla) en el esquema. Idealmente una de las reglas será la correcta. Comparado con la creación de reglas de mapeo lógicas, es más sencillo para los usuarios definir correspondencias simples entre los elementos atómicos de la base de datos y las propiedades de tipo de datos de las clases en la ontología, y posteriormente evaluar las formulas devueltas por MAPONTO.

En [Cullot et al., 2007] se presenta una herramienta para generar de forma automática ontologías a partir de un esquema de bases de datos. El proceso de mapeo comienza detectando casos particulares para elementos conceptuales en la base de datos, y transformándolos en los correspondientes componentes de la ontología.

El artículo presenta una herramienta llamada DB2OWL, la cual crea una ontología expresada en OWL-DL a partir de una base de datos. La aproximación que se emplea es la de la transformación de tablas a clases y atributos a propiedades. Se distinguen tres posible casos:

- **Caso 1**: Relaciones muchos a muchos. En este caso la tabla no genera una clase, pero las relaciones muchos a muchos que representa, se traducen a propiedades de objeto. Para una tabla T que relaciona dos tablas T_1 y T_2 que han sido traducidas en la ontología como las clases c_1 y c_2 respectivamente, se generan dos propiedades de objeto. Se define una propiedad op_1 con dominio c_1 y rango c_2, y otra propiedad op_2 con dominio c_2 y rango c_1. Las propiedades op_1 y op_2 son inversas la una respecto a la otra.

- **Caso 2**: Una tabla T se relaciona con otra tabla T_1 mediante una restricción de integridad referencial y los atributos locales que referencian además conforman una clave primaria. Las tablas en este caso se mapean como subclases de la clase asociada a la tabla con la que están relacionadas. Una tabla T relacionada con otra tabla T_1 mediante una clave externa que además es clave primaria, genera una clase c que es subclase de c_1, la clase generada a partir de T_1. Toda clave externa de la tabla T a una tabla T_n, que no sea clave primaria se mapea como una propiedad de objeto funcional de dominio c_1 y rango c_n, siendo estas las clases generadas a partir de las respectivas tablas. De forma adicional se define otra propiedad de objeto cuyo dominio es c_n y su rango c_1.

- **Caso 3**: Resto de casos no tratados en 1 y 2. Las tablas se mapean como clases. Las claves externas se tratan como en el caso anterior.

  Para toda tabla T que contenga atributos que no son claves externas, cada atributo se representa en la ontología como una propiedad de tipo de dato dp cuyo dominio es la clase c generada por la tabla T, y cuyo rango es el tipo de dato equivalente al del atributo en XML Schema.

El mapeo resultante entre los elementos del esquema y la ontología se almacena como un documento R2O.

2.1.4. Sistemas de Almacenamiento de Ontologías

En este apartado analizaremos algunos artículos que proponen distintas alternativas en cuanto al almacenamiento de ontologías en bases de datos. Existen dos grandes grupos en cuanto a las propuestas realizadas, aquellos trabajos que tratan con el almacenamiento de ontologías en bases de datos relacionales, y aquellos que tratan con bases de datos de propósito específico para el almacenamiento de ontologías.

Nos centraremos particularmente en el primer grupo de trabajos, ya que uno de los objetivos que nos hemos propuesto es el almacenamiento de la ontología en la base de datos junto con sus instancias. A pesar de esto es conveniente realizar un breve repaso a las soluciones de representación nativas, ya que pueden aportar ideas acerca de como afrontar los problemas que puedan surgir en la representación del almacenamiento.

Almacenamiento de Ontologías en Bases de Datos Relacionales

En esta sección realizaremos un breve repaso por los principales modelos e implementaciones para el almacenamiento de datos RDF en bases de datos relacionales.

Entre los principales esquemas de representación de ontologías RDFS en bases de datos relacionales se encuentran la representación vertical y binaria, así como una variante híbrida.

La representación vertical (schema-oblivious) consiste en el almacenamiento de los triples RDF en una única tabla con tres columnas, *Sujeto*, *predicado*, *Objeto*. Esta representación es sencilla de implementar y permite almacenar en una única tabla la información referente a la ontología y las instancias (TBox y ABox). Este tipo de representación tiene el inconveniente de que la consulta de las instancias de la ontología y el razonamiento, pueden ser muy costosos dadas las características de almacenamiento. Para mejorar el rendimiento habría que definir índices sobre las distintas columnas. Otra forma de mejorar el rendimiento es agrupar las tuplas atendiendo al valor de la columna *Predicado*, esto va a permitir almacenar en posiciones cercanas los elementos que potencialmente van a ser recuperados juntos por una consulta.

La representación vertical cuenta con algunas variantes, atendiendo principalmente a la forma en que se almacenan las URIs de los recursos. La variante conocida como **URI** almacena las URIs en la tabla vertical de triples, esto provoca la repetición de una misma URI en diferentes tuplas que representan triples. La otra variante conocida como **ID** consiste en crear una tabla adicional en la que se asocia a cada URI un identificador único. Como se puede imaginar la primera aproximación es ineficiente en cuanto al espacio de almacenamiento requerido, pero sin embargo la segunda lo es con respecto a la consulta, ya que debe hacerse una reunión adicional con la tabla de IDs para obtener la URI correspondiente.

Con el fin de evitar los inconvenientes de la aproximación por tabla vertical, se propuso un esquema dual, que separaba en diferentes tablas la ontología y las instancias. La estruc-

tura del almacenamiento de la ontología dependerá del modelo que se desee almacenar ya sea OWL o RDFS.

Otra variante es la representación binaria o específica (schema-aware) que consiste en la creación de una tabla, por cada propiedad o clase. Las tablas de propiedades tienen dos columnas *Sujeto* y *Objeto* y las de clases tienen una única columna con el *Sujeto* del triple (la instancia que pertenece a la clase).

Sobre el esquema binario se han realizado una serie de variantes atendiendo a la representación de la subsunción de relaciones y clases. La variante conocida como **ISA** utiliza para representar la subsunción la herencia de tablas tal y como se viene describiendo desde SQL99, el problema es que no todos los sistemas gestores de bases de datos implementan dicha parte del estándar. Por otra parte nos encontramos con la variante **NOISA** que almacena las tablas siguiendo el modelo relacional y evitando el uso de la característica de herencia de tablas.

El esquema híbrido es una combinación de los dos anteriores en el que las propiedades se almacenan en tablas con tres columnas *Sujeto*, *Predicado*, *Objeto* como en el caso de la representación vertical, pero en lugar de tener una única tabla para las propiedades, se tiene una tabla por cada tipo de valor en el rango, por ejemplo, en una tabla estarán todas las propiedades cuyo rango sea un entero, en otras aquellas cuyo rango sea una cadena, en otra aquellas que su rango sea un recurso de la ontología, etc... En el caso del almacenamiento de las clases y las instancias, se tiene una única tabla vertical, con dos columnas *Sujeto* y *Objeto*, donde el sujeto es una instancia, y el objeto la clase a la que pertenece.

La modificación de la estructura de la ontología (TBox) tiene diferentes efectos en cada una de las aproximaciones, en el modelo de tabla vertical la modificación de la ontología no tiene gran impacto, puesto que los cambios se reflejan con la modificación de las tuplas introducidas en la tabla. Sin embargo, en el esquema binario los cambios en las propiedades y clases implican la eliminación/creación de tablas en la base de datos. En el esquema híbrido el impacto de la evolución de la ontología es similar al primer caso, ya que tanto propiedades como clases se representan como tuplas de un conjunto de tablas.

Una diferencia bastante importante entre el modelo vertical y el binario, es la imposibilidad de reflejar tipos en el primero. Dado que todos los valores de las propiedades se almacenan en la misma columna, estos se almacenan como cadenas de texto, perdiendo así toda información sobre el tipo. En el caso del esquema binario cada propiedad puede definir la columna *Objeto* con el tipo correspondiente. El esquema híbrido al agrupar las propiedades atendiendo al tipo de su rango también dota de tipo a los valores correspondientes.

Así pues, podemos observar que el modelo híbrido soporta bien los cambios en la ontología y además tiene la ventaja de conservar los tipos de los diferentes valores.

Otra consideración a tener en cuenta es la posibilidad de realizar inferencia sobre las ontologías almacenadas en la base de datos, para ello, hay dos posibilidades atendiendo al momento en que se realiza la inferencia. Dependiendo de si el cálculo de los nuevos triples inferidos se realiza en tiempo de compilación o en tiempo de ejecución hablaremos de aproximaciones *a priori* o *a posteriori*. En caso de realizarse un cálculo *a priori* se evita tener que hacer los cálculos del cierre transitivo cada vez que se realiza una consulta en el sistema. Sin embargo nos encontramos con el inconveniente de que se emplea más

espacio de almacenamiento, y también se hace más dificultoso el proceso de actualización en caso de haber modificaciones en la ontología. En el caso de la ejecución *a posteriori* los requisitos de almacenamiento son menores, pero es necesario tener suficiente espacio de almacenamiento en memoria principal para poder realizar el cálculo del cierre transitivo.

Las propuestas que utilizan el esquema de tabla vertical suelen realizar el cálculo *a priori*, aunque existen otras alternativas que muestran los triples inferidos como vistas materializadas. En este caso la vista materializada contendrá las instancias inferidas, así como las originales que contuviese la clase. La vista se crea de forma recursiva como la unión de las instancias pertenecientes a una clase con las vistas correspondientes con sus subclases.

Por otra parte, los esquemas binario e híbrido utilizan una inferencia a posteriori o en tiempo de ejecución. En el caso de los modelos ISA e Híbrido utilizan una codificación interna para la subsunción en base a etiquetas de clases y propiedades basadas en intervalos. Este tipo de codificación permite sustituir un cálculo del cierre transitivo costoso por consultas de rango apropiadas, reduciendo así las necesidades en cuanto a memoria principal.

A tenor de los experimentos y evaluaciones llevados a cabo en [Theoharis et al., 2005] el modelo híbrido y el de vistas materializadas presentan mejor rendimiento en cuanto a tiempo de ejecución de las consultas. El rendimiento de ambas aproximaciones es similar independientemente del tamaño del conjunto de datos, sin embargo el modelo de vistas materializadas se comporta un poco mejor en casos en los que las clases consultadas se encuentran en niveles intermedios de la jerarquía. Esto se debe a que el modelo de vistas materializadas tiene la subsunción precalculada, y por tanto las consultas son más rápidas. Sin embargo hay que tener en cuenta el coste en cuanto al espacio de almacenamiento y la complejidad de actualización de las vistas materializadas.

RDFSuite [Alexaki et al., 2001] almacena ontologías RDFs y sus instancias para consulta. Tanto la ontología como las instancias se almacenan en una base de datos objeto relacional. La estructura de la ontología RDFs (TBox) se almacena en cuatro tablas *Class*, *Property*, *Subclass* y *Subproperty*, almacenando las clases, las propiedades, las relaciones de subclase y de subpropiedad respectivamente. Para ahorrar espacio de almacenamiento en el nombre de las clases y propiedades, se considera una tabla *Namespace* donde se almacenan los espacios de nombres de cada uno de los esquemas RDF almacenados. Además existe una tabla *Type* donde se almacenan los nombres de los tipos RDFs. Las instancias RDF se mantienen en tablas separadas. RDFSuite soporta el lenguaje de consulta RQL. RDFSuite está diseñada para trabajar con RDF y RDFs por lo que no es compatible con los nuevos lenguajes de definición de ontologías tales como OWL, por tanto, su capacidad expresiva es limitada.

Sesame [Broekstra et al., 2002, Broekstra et al., 2003]es una arquitectura basada en Web para el almacenamiento eficiente y la consulta expresiva de grandes cantidades de metadatos RDF.

Los datos RDF se almacenan en un sistema gestor de base de datos. Para evitar la dependencia de un SGBD concreto, toda la implementación de acceso a un SGBD concreto

se ubica en una capa de la arquitectura conocida como SAIL (Storage And Inference Layer) y encargada de la gestión del almacenamiento y la inferencia.

SAIL es un interfaz de programación (API) que proporciona métodos de acceso específicos a RDF y transforma dichos métodos en llamadas a un SGBD específico. La importancia de la capa SAIL reside en la posibilidad de implementar Sesame sobre una amplia variedad de repositorios sin tener que modificar otros componentes de la arquitectura.

Sesame proporciona diferentes implementaciones de la API SAIL, dependiendo del tipo de repositorio empleado.

Los módulos funcionales de Sesame son clientes de SAIL. Estos componentes son los siguientes:

- **The RQL query engine:** Evalúa consultas RQL realizadas por los usuarios. Las consultas RQL se transforman en un conjunto de llamadas a la API SAIL.
- **The RDF admin module:** Permite la carga incremental de datos RDF y de información del esquema, así como el borrado de información. Además este módulo comprueba la consistencia de cada nueva sentencia incluida con el resto de información existente en el repositorio, infiriendo nueva información en caso de ser necesario.
- **The RDF export module:** Permite la extracción del esquema o de los datos de un modelo en formato RDF.

La comunicación entre los diferentes módulos de Sesame se realiza utilizando métodos apropiados al entorno. En un contexto Web la comunicación se realizará sobre HTTP, sin embargo en otros contextos puede ser más apropiado el uso de Remote Method Invocation (RMI) o Simple Object Access Protocol (SOAP).

El repositorio en el que se almacenan los datos RDF puede ser un SGBD, bases de datos específicas para RDF, ficheros RDF o servicios de red. Dependiendo del tipo de repositorio empleado deberemos utilizar su API SAIL correspondiente.

En [Horrocks et al., 2004] se presenta la aplicación Instance Store (iS) cuyo objetivo es resolver los problemas de escalabilidad que aparecen cuando se realiza razonamiento sobre ontologías con una gran cantidad de individuos. Para resolver este problema se combina un razonador sobre lógicas descriptivas con una base de datos.

El sistema parte de la suposición de la no existencia de roles en la ABox, es decir, no existen relaciones entre instancias. Por tanto, bajo esta suposición es posible reducir el razonamiento sobre instancias a un razonamiento sobre sus descripciones.

El desarrollo del sistema se encuentra estancado desde 2005. La última versión estable del Software data de Agosto de 2004.

En [Li, 2004] se presenta una extensión de Instance Store mediante la cual se evita la restricción por la cual no era posible expresar roles entre instancias. Utilizando una técnica para DL llamada *precompletion* es posible transformar una ABox general conteniendo aserciones con roles en una equivalente sin roles. Esto se consigue expresando en forma de aserciones de concepto adicionales las aserciones sobre roles. El conjunto de reglas obtenidas

no es determinista, por lo que tras ser procesada, una misma ABox puede generar diferentes ABoxes sin roles.

Preprocesando los datos en la ABox empleando esta técnica, se obtienen ABoxes que pueden ser tratadas mediante Instance Store.

DLDB [Pan and Heflin, 2003] es un sistema que extiende un sistema gestor de bases de datos relacional con capacidades adicionales que permiten el razonamiento con DAML+OIL. El prototipo implementado utiliza MS Access como base de datos y FaCT DL como motor de inferencia.

El sistema utiliza un sistema híbrido que combina dos métodos de almacenamiento de RDF, tabla de propiedades y clase horizontal. El método de tabla de propiedades consiste en generar en la base de datos una tabla por cada una de las propiedades en la ontología. Por su parte, el de clase horizontal consiste en la generación de una tabla en la base de datos por cada clase en la ontología. Este método híbrido permite la representación de clases y propiedades de la ontología como tablas en la base de datos. Las tablas de propiedades contendrán referencias a las tablas de clase que representan su dominio y rango. Los identificadores de las clases y propiedades, actuarán como nombre en las tablas generadas.

Este modelo de almacenamiento tiene la desventaja de no escalar bien cuando el número de clases y propiedades supera el centenar. Sin embargo, este tipo de casos representan un número poco representativo con respecto al total, siendo más bien excepcionales.

La implicación RDF en DAML+OIL es relativamente simple, consistiendo en inferencia taxonómica usando *rdfs:subClassOf* y *rdfs:type*, y de forma similar para *rdfs:subPropertyOf*.

El sistema almacena la información de la jerarquía de clases a través de vistas. Dichas vistas se definen de forma recursiva, de tal forma que una vista de clase contiene la información de la tabla de la clase y la vista de sus subclases directas. De este modo la vista incluye las instancias indicadas explícitamente y aquellas que han sido inferidas. Desde un punto de vista de bases de datos deductivas, podría decirse que las vistas son relaciones intensionales definidas por reglas lógicas (consulta) y que las tablas son relaciones extensionales que almacenan la información explícita.

Es posible utilizar un razonador DL para precalcular la subsunción de clases y a partir de la taxonomía inferida, crear las correspondientes vistas.

La API de DLDB soporta consultas conjuntivas en un formato similar a KIF. Durante la ejecución las consultas se transforman en sentencias SQL estándar y se envían a la base de datos usando JDBC. El SGBD procesa la consulta SQL y devuelve los correspondientes resultados.

En 2008 se presentó una nueva versión de DLDB denominada DLDB2 [Pan et al., 2008]. Entre las novedades que se encuentran en esta versión está el que trabaja con OWL y soporta inferencias para un subconjunto de los elementos empleados en OWL-DL. El modelo de arquitectura ha sido generalizado y ahora cualquier razonador con capacidad DIG y cualquier sistemas gestor de bases de datos que soporte el acceso mediante JDBC pueden emplearse junto a DLDB2. Cuando se realiza la carga de una ontología se separan los datos de la TBox de los datos de instancia de la ABox. Empleando el razonador DL se realiza un proceso de razonamiento sobre la TBox y los resultados se utilizan para crear tablas

y vistas en la base de datos. Los datos de instancia se cargan directamente en las tablas correspondientes.

3Store [Harris and Gibbins, 2003] es un sistema de almacenamiento y consulta sobre datos RDF desarrollado en la Universidad de Southampton. El sistema se diseñó para proporcionar interfaces adecuadas, razonamiento y eficiencia en la gestión de cantidades masivas de datos RDF y RDFS. 3Store utiliza MySQL como sistema gestor de bases de datos. El esquema de almacenamiento para 3Store está compuesto por cuatro tablas *triples*, *models*, *resources* y *literals*. Toda la información está centralizada en la tabla *triples* donde se almacenan el sujeto, objeto y predicado de los triples RDF. El esquema está normalizado, de modo que se utilizan identificadores para representar los modelos, recursos y literales, los cuales hacen referencia a las correspondientes tablas donde estos se almacenan. La tabla triples contiene los campos *model*, *subject*, *predicate*, *object*, *literal*, *inferred*. Los cuatro primeros son identificadores de elementos situados en las otras tablas, mientras que los dos últimos son valores binarios. Para el campo *literal* un valor de verdad indica que el objeto del triple es un literal, y un valor falso indica que es un recurso, y que por tanto ha de ser buscado en la tabla correspondiente. El campo *inferred* indica si el triple ha sido introducido en el sistema o si ha sido inferido a partir de información que se encontraba almacenada previamente.

3Store implementa un motor RDQL que transforma las consultas sobre RDF recibidas en consultas SQL que envía al motor de bases de datos subyacente. En lo que concierne al razonamiento se utiliza un método híbrido en el que el razonamiento más sencillo y que genera un número de triples adicional menor se realiza de forma anticipada, y el razonamiento más complejo y con mayor necesidad de almacenamiento se resuelve en tiempo de consulta.

MINERVA [Zhou et al., 2006] es un sistema de consulta, almacenamiento e inferencia para ontologías OWL de gran tamaño, construido sobre una base de datos relacional. MINERVA está desarrollado por IBM, y se distribuye como un componente del Integrated Ontology Development Toolkit (IODT) de IBM.

En tiempo de ejecución de consultas, MINERVA soporta Description Logic Programs (DLP), que pueden verse como un subconjunto del lenguaje SPARQL y que permite la ejecución de OWL-DL.

El sistema combina razonadores de Lógica Descriptiva para inferencia en la TBox con reglas lógicas extraídas de los DLP para la inferencia en la ABox. Se adapta el esquema de la base de datos en base a los requisitos de inferencia. Las consultas de los usuarios se responden recuperando directamente los resultados de la base de datos.

OntoDB [Dehainsala et al., 2007] es una base de datos basada en ontologías (ontology-based database (OBDB)) implementada sobre un sistema gestor de bases de datos relacional. Se propone una nueva representación para el almacenamiento de ontología en bases de datos, descrita como *tabla por clase*. Esta representación consiste en crear una tabla en la base de datos por cada clase en la ontología, adicionalmente los valores de las propiedades de la clase se representan como columnas de dicha tabla.

OntoDB presenta una arquitectura en la que se distinguen cuatro partes:

1. Datos : Es la parte del sistema gestor de bases de datos donde se almacenan los datos.
2. Metadatos : Metadatos del SGBD, compuesto por las tablas del catálogo del sistema.
3. Ontología : Se define mediante un mapeo objeto-relacional.
4. Meta-Schema : Registra el modelo de la ontología en forma de meta-modelo reflexivo.

OntoDB puede de forma explícita, ontologías, datos del esquema, los datos sobre instancias y el enlace entre los datos y la ontología.

OpenLink Virtuoso [Erling and Mikhailov, 2009] es un servidor comercial multi-protocolo que proporciona acceso a datos relacionales almacenados dentro del propio servidor o en una combinación de bases de datos relacionales externas. Virtuoso almacena RDF y soporta consultas SPARQL. El modelo de almacenamiento es híbrido, se basa en una tabla de triples vertical, y un conjunto de tablas auxiliares para almacenar los tipos. La tabla de triples (en realidad quads, puesto que se almacena la URI del grafo al que pertenece el triple) contiene la mayoría de la información en forma de identificadores, y se hace uso de las tablas adicionales cuando se han de representar tipos mas complejos.

Sistemas de Propósito Específico para el Almacenamiento de Ontologías

Uno de los primeros sistemas para la gestión de ontología de gran tamaño fue PARKA-DB [Stoffel et al., 1997]. PARKA era un sistema basado en *frames* que utilizaba una pequeña base de datos relacional diseñada específicamente para dar soporte a la herramienta. Parka constaba de una arquitectura en 3 niveles, en el más bajo se encontraba la base de datos que se encargaba de realizar tareas básica de gestión de datos, en el segundo nivel se encontraban una serie de algoritmos de inferencia eficientes que interactuaban con el primer nivel, el tercer nivel contenía una interfaz de usuario de propósito general. Existe una versión denominada PARKA-SW que fue adaptada para soportar RDF.

RDFDB [Guha, 2000] es una base de datos simple, escalable y de código abierto, para el almacenamiento de RDF. RDFDB usa Berkeley DB(BDB)/Sleepycat DB como servidor de almacenamiento. Organiza tres bases de datos B-Tree (*arcindex.db*, *findex.db*, *rindex.db*) de modo que dada una sentencia RDF se almacena en disco como tres pares de datos (uno en cada base de datos). *findex.db* indexa los datos por el hash compuesto por la concatenación del Sujeto y el Predicado de la sentencia RDF, del mismo modo *rindex.db* se indexa por la concatenación del Objeto y el Predicado, y finalmente *arcindex.db* se indexa por el Predicado. Dependiendo del tipo de consulta se consultará un hash determinado.

RDFDB está considerada como la primera base de datos RDF. Actualmente su desarrollo está detenido y las últimas modificaciones datan de 2005.

Redland [Beckett, 2002] es una implementación extensible y eficiente de RDF, que proporciona interfaces de alto nivel para la manipulación consulta y almacenamiento de instancias del modelo en C, Perl, Python, Tcl y otros lenguajes. La implementación del almacenamiento en Redland utiliza múltiples Hashes para almacenar las sentencias, de

forma similar a como lo hace RDFDB. Un Hash en Redland es el mapeo de una clave a un valor, permitiendo duplicados. Concretamente la implementación almacena cada sentencia RDF como tres hashes SP2O, PO2S y SO2P. Cada uno de estos hashes supone una combinación diferente del triple Sujeto-Predicado-Objeto, con el objetivo de acelerar el tiempo de respuesta ante una consulta. Las combinaciones se han elegido atendiendo a criterios experimentales realizados por los autores.

Los hashes pueden almacenarse de forma persistente, en este caso Redland emplea como base de datos Berkeley DB(BDB)/Sleepycat DB. BDB proporciona utilidades para la persistencia de objetos para diferentes lenguajes de programación.

TAP es la implementación de la consulta y las interfaces/protocolos de negociación definidos en [Guha and McCool, 2003]. TAP proporciona mecanismos para publicar datos mediante TAPache, un sistema escrito en C como un módulo de Apache que implementa el almacenamiento y consulta de datos RDF. TAP usa una interfaz de consulta sencilla llamado GetData, que permite consultar grafos RDF. Soporta las propiedades de subclase y subpropiedad de RDF Schema. La gestión del almacenamiento en TAP se delega en sistemas gestores de bases de datos como MySQL o BerkeleyDB. El desarrollo de este sistema fue realizado en la Universidad de Stanford. El principal problema de este sistema es que fue diseñado con el objetivo de facilitar la publicación de datos semánticos en la web, pero no es un sistema que permita gestionar grandes cantidades de datos y tráfico de red.

Kowari [Wood et al., 2005] es un proyecto de código abierto, patrocinado por Tucana Technologies y se licencia bajo la licencia pública de Mozilla v1.1. El sistema Kowari Metastore es una base de propósito específico para el almacenamiento de RDF. Kowari proporciona una infraestrucrura escalable para el almacenamiento de sentencias RDF, así como asegura la seguridad de las transacciones y proporciona un lenguaje de consulta específico basado en RDFS y OWL, iTQL (interactive Tucana Query Language). Este lenguaje soporta la expresividad de OWL Lite mas las restricciones de cardinalidad de OWL Full.

El formato de almacenamiento nativo de Kowari y su modelo de indexación permiten escalar el sistema hasta alcanzar la cifra de millones de sentencias RDF almacenadas en una única máquina.

El almacén de datos nativo de Kowari (XA Statement Store), es un almacén implementado en Java 1.4 que permite el almacenamiento persistente de sentencias en disco, sin utilizar una base de datos externa. El almacenamiento de cada sentencia en la *Statement Store* consta de cuatro nodos sujeto, predicado y objeto, correspondientes a la sentencia RDF, y un nodo adicional que describe el modelo al que pertenece la sentencia. Para incrementar la velocidad y reducir la redundancia, cada nodo se representa con un identificador que es un entero de 64 bits. Estos valores se encuentran en el *Node Pool* un almacén de nodos. Los valores numéricos corresponden con URIs y literales que se encuentran en otro almacén conocido como *String Pool*. Ambos almacenes se complementan y proporcionan todos los requisitos necesarios para el almacenamiento de RDF. Sobre los almacenes de datos se construyen índices para mejorar el rendimiento de las consultas, se emplea una

combinación de árboles-B y árboles Adelson-Velskii and Landis(AVL).

Kowari implementa almacenamiento basado en Jena, lo que permite utilizar un subconjunto de RDQL como alternativa al lenguaje de consulta iTQL.

Posteriomente en [Wood, 2005] se introducen mejoras en el sistema, modificando el modelo de almacenamiento de los nodos pasando del modelo transaccional a uno basado en *skip lists*. Esto permite mejorar el rendimiento y tiempo de respuesta de las consultas. También se ha incluido la capacidad de distribuir consultas entre varias máquinas, apoyando esta propuesta se ha desarrollado un sistema de resolución de fuentes, que permite conocer los modelos que contiene cada almacén, y por tanto, si puede o no participar en la resolución de la consulta. Si un almacén no almacena el modelo sobre el que se consulta, no se conecta con él y se ahorran recursos. Para mejorar el rendimiento también se contempla la posibilidad de ejecutar varias instancias de Kowari en una misma máquina.

Las primeras versiones de Kowari sólo permitían la consulta, pero no era posible razonar sobre los datos. Para dotar de inferencia al sistema se creó Krule un motor de inferencia especialmente diseñado para Kowari. Las reglas de Krule están escritas en RDF, de acuerdo a una ontología OWL que define su estructura. Krule aplica reglas de implicación a un conjunto base de sentencias RDF y produce nuevos datos. Estos datos se almacenan en el mismo modelo RDF que los generó o en uno nuevo.

El proyecto dejó de estar activo a finales de 2006, y se creó un proyecto paralelo de código abierto llamado Mulgara que partiendo de la última versión de Kowari continúa en desarrollo actualmente.

2.2. Extracción de Datos Semánticos a partir de Texto No Estructurado

En esta sección realizaremos una revisión del estado del arte de las investigaciones relacionadas con la extracción de datos semánticos a partir de texto no estructurado.

2.2.1. Aprendizaje de Ontologías a partir de Textos

En este apartado trataremos sobre el aprendizaje de ontologías, particularmente nos centraremos en el aprendizaje de ontologías a partir de texto. Aunque formalmente las ontologías especifican un modelo de un dominio, la parte extensional la proporciona una base de conocimiento que contiene aserciones sobre instancias de conceptos y relaciones definidos en base a la ontología. El soporte automático de la construcción de una ontología es lo que se conoce como aprendizaje de ontologías.

Las primeras referencias al término aprendizaje de ontologías (Ontology Learning) podemos encontrarlas en [Madche and Staab, 2001] donde se describe en términos de la adquisición de un modelo de un domino a través de los datos.

El proceso de aprendizaje de ontologías necesita datos de entrada a partir de los cuales aprender los conceptos, definiciones y relaciones relevantes para un determinado dominio. El requisito indispensable es que los datos de entrada sean relevantes al dominio del cual queremos aprender la ontología. Los datos de entrada para un proceso de aprendizaje de ontologías pueden ser muy variados, desde descripciones de esquemas como esquemas XML y DTDs, a diagramas UML y esquemas de bases de datos. El aprendizaje de ontologías también se puede realizar sobre fuentes semiestructuradas como documentos XML y páginas HTML. Cuando el aprendizaje de ontologías se realiza sobre fuentes textuales no estructuradas, es cuando hablamos de aprendizaje de ontologías a partir de texto.

El proceso de aprendizaje de ontologías puede verse como un proceso de ingeniería inversa, en el que un autor plasma información sobre un dominio de discurso en forma de texto, y a partir de dicho texto, reconstruimos el modelo original que dicho autor plasmó en palabras. Esta tarea entraña una gran complejidad debido principalmente a dos razones. En primer lugar, en el proceso de creación de un texto sólo refleja el dominio de conocimiento del autor de forma parcial, por lo que el proceso de ingeniería inversa como mucho podrá reconstruir parcialmente dicho modelo de conocimiento. En segundo lugar y mucho más importante, el conocimiento del mundo (a no ser que consideremos un libro de texto o diccionario) raramente se menciona de forma explícita. El aprendizaje de ontologías a partir de texto, trata con símbolos no interpretados y signos para los cuales debe identificarse su significado apropiado.

El proceso de aprender las extensiones para conceptos y relaciones es a lo que comúnmente se le denomina población de ontologías (ontology population). También se habla de anotación del conocimiento si el proceso de población se realiza seleccionando fragmentos textuales de un documento y se asignan a conceptos ontológicos.

En los últimos años se han desarrollado una amplia variedad de métodos para aprendizaje de ontologías en textos, por desgracia no existe un consenso entre la comunidad investigadora en cuanto a cuales son las tareas concretas que deben realizarse en el proceso

de aprendizaje. En nuestro caso nos ceñiremos a la definición de subtareas para el desarrollo de ontologías definida en [Cimiano, 2006]. La tarea de desarrollo de ontologías se basa principalmente en la definición de conceptos y las relaciones entre estos. En algunas aplicaciones de ontologías en minería de textos, o simplemente para facilitar la interpretación de las ontologías por parte de humanos, es importante conectar los conceptos y las relaciones con los símbolos que se utilizan para hacer referencia a ellos. En este caso concreto implica la adquisición de conocimiento lingüístico acerca de los términos que se utilizan para referirse a un concepto específico y sinónimos potenciales de estos términos. Una ontología está compuesta por una jerarquía de conceptos, así como de otras relaciones de naturaleza no jerárquica. Se puede instanciar información adicional para los conceptos y relaciones con el fin de restringir su interpretación. Por último, las ontologías también se usan para derivar conocimiento no explícito, para conseguir este propósito se deben obtener axiomas lógicos que modelen implicaciones entre conceptos y relaciones. Todas las primitivas ontológicas se pueden organizar en un esquema de complejidad incremental, atendiendo al coste de adquirir dichas primitivas en cada una de las subtareas de aprendizaje de ontologías.

En la mayoría de los casos las capas se construyen conceptualmente unas sobre otras, dependiendo los procesos en capas superiores de las salidas de procesos situados en capas inferiores. Sin embargo, desde el punto de vista del procesamiento, las capas pueden agruparse y sus funciones pueden ser realizadas por el mismo algoritmo. La figura 2.1 obtenida de [Cimiano, 2006] muestra la estructura general de las distintas capas y presenta un breve ejemplo acerca del conocimiento a aprender en cada una de ellas.

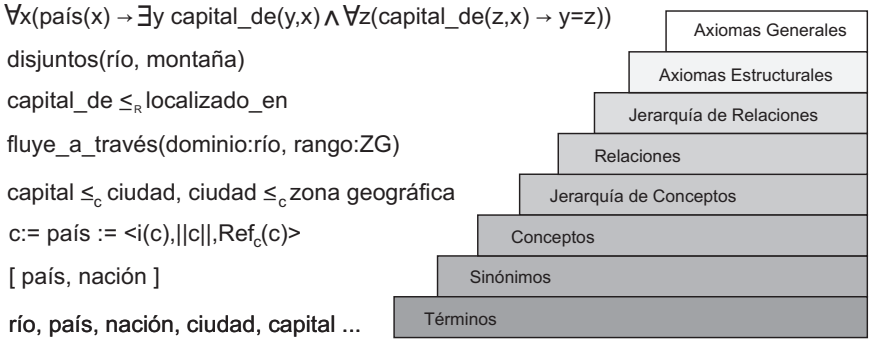


Figura 2.1: Estructura por capas del aprendizaje de Ontologías

Las tareas definidas en este esquema en capas son:

- Adquisición de la **terminología** relevante.

  La extracción de términos es un requisito fundamental en todos los aspectos del aprendizaje de ontologías desde textos. Los términos son la representación lingüística de conceptos de un dominio específico y por tanto son esenciales para posteriores tareas más complejas. La tarea a realizar en esta capa es la identificación de un conjunto relevante de términos que identifiquen conceptos o relaciones. Desde un punto de vista lingüístico un término puede estar compuesto por una o más palabras, por lo general los términos formados por más de una palabra suelen tener un significado

técnico en un contexto o dominio concreto. La tarea de identificación de términos tendrá como entrada un conjunto de textos acerca de un dominio de interés, y producirá una salida compuesta por un conjunto de cadenas que representan términos asociados a conceptos o relaciones.

Existen diversos métodos para la extracción de términos, entre los más conocidos existen dos corrientes diferenciadas, la que aborda el problema desde un punto de vista estadístico de recuperación de información, particularmente conocido el trabajo de [Salton and Buckley, 1988], y otra que está más orientada al procesamiento del lenguaje natural.

La identificación de término implica la identificación de estructuras gramaticales que permitan determinar la función de los términos en el contexto de la frase. El procesamiento de las frases se realiza mediante etiquetado de categoría gramatical (Part-of-Speech). Una vez extraídos y etiquetados los términos se puede proceder a la siguiente etapa en la que se comienza a dotar a estos de cierta semántica.

- Identificación de término **sinónimos**.

  La tarea de la identificación de sinónimos consiste en encontrar palabras que denotan el mismo concepto a pesar de que su representación sintáctica sea distinta. Por otra parte, es cierto que raramente existen sinónimos reales, en el sentido de que diferentes palabras consideradas sinónimos tienen ciertas connotaciones propias que dependen del contexto en el que se encuentran. En este caso nos ceñiremos a la definición que se hace sobre conjuntos de sinónimos en WordNet, los llamados *synsets* [Fellbaum, 1998]. Consideraremos sinónimos todas aquellos términos que aparezcan dentro del mismo *synset* de WordNet.

  En este campo las técnicas de reducción de dimensionalidad tales como LSI (Latent Semantic Indexing) [Landauer and Dumais, 1997] y PLSI (Probabilistic Latent Semantic Indexing) [Hofmann, 1999] han demostrado obtener buenos resultados. Desde el punto de vista estadístico se han empleado técnicas de agrupamiento para encontrar términos relacionados.

  En [Turney, 2001] mediante un algoritmo no supervisado se extraen sinónimos basándose en criterios de tipo estadístico. El método consiste en determinar la similitud entre términos, considerando los más similares sinónimos. Se ha evaluado con ejercicios de identificación de sinónimos en los exámenes de inglés TOEFL con una aceptable grado de éxito.

- Formación de **conceptos**.

  La formación de conceptos debe proporcionar una definición intensional de los conceptos, su extensión y los símbolos léxicos utilizados para referirse a ellos. Según [Cimiano, 2006] un concepto debe definirse como un triple $< i(c), [[c]], Ref_C(c) >$ donde $i(c)$ es la descripción intensional del concepto, $[[c]]$ es la extensión y $Ref_C(c)$ describe su realización léxica en un corpus. Se asume que la definición intensional es un descripción en lenguaje natural de forma parecida a las glosas de WordNet.

La detección de conceptos es algo más complejo puesto que podemos entender la idea de concepto de formas diversas. Desde el punto de vista del agrupamiento, un concepto estaría formado por conjuntos de términos relacionados. Pero también podemos pensar que varios términos relacionados forman por sí mismos conceptos que comparten una relación.

- Organización jerárquica de los conceptos **(jerarquía conceptual)**.

  A continuación nos centramos en tareas dedicadas a inducir, extender y refinar la ontología para obtener una jerarquía de conceptos.

  El proceso comienza con la inducción de la jerarquía a partir de los conceptos identificados en la capa anterior. Se continúa con un refinamiento en el que se extiende la jerarquía conceptual existente con subconceptos adicionales o conceptos ya existentes. Finalmente se realiza un proceso de extensión léxica en el cual dado un concepto y su realización léxica, se añaden nuevos términos que se adapten a la definición.

  Actualmente existen tres paradigmas esenciales para inducir jerarquías de conceptos a partir de textos, aplicación de patrones léxico-sintácticos indicando la relación de interés (en la línea del trabajo [Hearst, 1992]), basados en la hipótesis de distribución de Harris, donde se emplean algoritmos de agrupamiento jerárquico para derivar jerarquías de conceptos de forma automática a partir del texto y basado en la coocurrencia de términos en la misma frase, párrafo o documento.

  También es habitual el recurrir a ontologías externas y relacionar los textos con los conceptos previamente diseñados por expertos. Particularmente está muy extendido el uso de WordNet y Wikipedia como ontologías de referencia.

- Aprendizaje de **relaciones**, propiedades o atributos, junto con sus dominios y rangos apropiados.

  En este caso el aprendizaje de relaciones se restringe a relaciones binarias, identificadas por una etiqueta, y de las que también se quiere aprender su dominio y rango.

- Organización jerárquica de las relaciones (jerarquía de relaciones).

  Consiste en una jerarquización adecuada de las relaciones aprendidas en la etapa anterior.

- Instanciación de los **axiomas del esquema**.

  Partiendo de axiomas estructurales tales como equivalencia o disyunción para conceptos, o transitividad, simetría etc... para las relaciones, se trata de encontrar las relaciones o pares de conceptos a los que se aplican esos axiomas.

- Definición de **axiomas generales**.

  La situación es diferente para el aprendizaje de axiomas generales, puesto que estos axiomas deben ser aprendidos en lugar de ser simplemente instanciados. Los axiomas generales pueden verse como implicaciones lógicas que restringen la interpretación de

los conceptos y relaciones. Se diferencia de los axiomas estructurales en que no aparecen tan habitualmente. El aprendizaje de axiomas puede verse como la derivación de conexiones y relaciones mas complejas entre los conceptos y las relaciones.

En nuestro caso el procedimiento que vamos a emplear nos permitirá llegar hasta el nivel de organización jerárquica de los conceptos. Es por esto que no hemos hecho especial énfasis en describir técnicas para el resto de niveles superiores.

2.2.2. Conjuntos-AP (AP-SETS)

En este apartado reseñaremos una representación intermedia para datos textuales definida en [Martínez-Folgoso, 2008] y que se obtiene por medio de la aplicación del algoritmo Apriori. Mediante la aplicación de esta técnica se obtiene un retículo de términos maximales que puede verse como una ontología de inclusión básica. Esta ontología será la base a partir de la cual realizaremos tareas de extensión semántica con el objetivo de dotarla de mayor significado.

Introducción

La información en las bases de datos relacionales, está altamente estructurada y permite un fácil acceso, simplificando las tareas de consulta. Los campos textuales de una base de datos pueden contener información altamente relevante para el usuario, pero no es fácil manejarla dado que estos campos carecen de estructura. Las operaciones que se pueden realizar sobre campos de texto son muy limitadas y no permiten extraer todo el conocimiento que se encierra en la representación textual. Únicamente podemos realizar operaciones de consulta a través de palabras o expresiones regulares que encajen con el contenido de los campos textuales.

Cuando es necesario realizar un proceso de resumen sobre los campos textuales, las operaciones estándar no son de ayuda mas allá de la posibilidad de realizar un conteo de palabras o de obtener frecuencias de aparición.

En [Marín et al., 2006, Martín-Bautista et al., 2008, Martín-Bautista et al., 2006] se propone la transformación de los campos textuales de una base de datos en una estructura intermedia que permita representar de forma más adecuada dichos textos. Dicha representación está basada en la representación de itemsets frecuentes, obtenidos mediante la aplicación del algoritmo Apriori [Agrawal and R.Srikant, 1994]. Esta representación intermedia es lo que se denomina Conjunto-AP. Los Conjuntos-AP se representan como un tipo de dato abstracto (TDA) en la base de datos y proporcionan un conjunto de operaciones para trabajar con ellos.
Formalmente un Conjunto-AP se define:

Definición Sea $X = \{x_1, ..., x_n\}$ cualquier referencial y $\mathcal{R} \subseteq \mathcal{P}(X)$ se dice que $\mathcal{R}$ es un Conjunto-AP si sólo si:
1. $\forall Z \in \mathcal{R} \Rightarrow \mathcal{P}(Z) \subseteq \mathcal{R}$
2. $\exists Y \in \mathcal{R}$ tal que:
(a) $card(Y) = max_{Z \in \mathcal{R}}(card(Z))$ and $\neg \exists Y' \in R | card(Y') = card(Y)$

(b) $\forall Z \in \mathcal{R}; Z \subseteq Y$

Según esta definición, para todo conjunto del Conjunto-AP sus componentes también deben ser subconjuntos del Conjunto-AP. Además existe un conjunto del Conjunto-AP tal que su cardinalidad es mayor que la del resto de conjuntos del Conjunto-AP, y todo conjunto que pertenece al Conjunto-AP es un subconjunto del conjunto de máxima cardinalidad. El conjunto Y de máxima cardinalidad caracteriza el Conjunto-AP y recibe el nombre de *conjunto de expansión de* $\mathcal{R}$. Se denota $\mathcal{R} = g(Y)$ al Conjunto-AP con conjunto de expansión Y. El *Nivel* de $g(Y)$ es la cardinalidad de Y.
Los Conjuntos-AP de nivel 1, serán los elementos de X y se considera el conjunto vacío $\emptyset$ el Conjunto-AP de nivel 0. Según esta definición un Conjunto-AP $g(Y)$ es un retículo de $\mathcal{P}(Y)$.
Adicionalmente se definen las siguientes operaciones:

Inclusión de Conjuntos-AP:

Definición Sean $\mathcal{R} = g(R)$ y $\mathcal{S} = g(S)$ dos Conjuntos-AP con el mismo referencial:
$\mathcal{R} \subseteq \mathcal{S} \Leftrightarrow R \subseteq S$

La inclusión de Conjuntos-AP se define como la inclusión de sus conjuntos de expansión. Un Conjunto-AP $\mathcal{R}$ está incluido en otro $\mathcal{S}$, si sólo si su conjunto de expansión R es un subconjunto del conjunto de expansión S del otro Conjunto-AP.

Sub-Conjunto-AP inducido:

Definición Sean $\mathcal{R} = g(R)$ y $Y \subseteq X$ se dice que $\mathcal{S}$ es el sub-Conjunto-AP inducido por Y si sólo si: $\mathcal{S} = g(R \bigcap Y)$

Dado un Conjunto-AP $\mathcal{R}$ y un conjunto Y que es subconjunto del referencial X. Se dice que $\mathcal{S}$ es el sub-Conjunto-AP de $\mathcal{R}$ inducido por Y, si sólo si $\mathcal{S}$ es el Conjunto-AP generado, cuyo conjunto de expansión es la intersección del conjunto de expansión R de $\mathcal{R}$ y el conjunto Y.

Super-Conjunto-AP inducido:

Definición Sean $\mathcal{R} = g(R)$ y $Y \subseteq X$ se dice que $\mathcal{V}$ es el sub-Conjunto-AP inducido por Y si sólo si: $\mathcal{V} = g(R \bigcup Y)$

Dado un Conjunto-AP $\mathcal{R}$ y un conjunto Y que es subconjunto del referencial X. Se dice que $\mathcal{V}$ es el super-Conjunto-AP de $\mathcal{R}$ inducido por Y, si sólo si $\mathcal{V}$ es el Conjunto-AP generado, cuyo conjunto de expansión es la unión del conjunto de expansión R de $\mathcal{R}$ y el conjunto Y.

Estructuras-AP (AP-Structures)

Una vez establecido el concepto de Conjunto-AP se definen las Estructuras-AP como conjuntos de Conjuntos-AP. Las Estructuras-AP se obtienen de forma incremental desde el conjunto de itemsets frecuentes de cardinalidad 1 hasta obtener el itemset frecuente de cardinalidad maximal, todo ello obtenido sobre un valor mínimo de soporte. Formalmente:

Definición Sea $X = \{x_1, ..., x_n\}$ cualquier referencial y $S = \{A, B, ...\} \subseteq \mathcal{P}(X)$ tal que:
$\forall A, B \in S; A \nsubseteq B, B \nsubseteq A$
Se llama Estructura-AP de expansión S, $\mathcal{T} = g(A, B, ...)$ al conjunto de Conjuntos-AP cuyos conjuntos de expansión son $A, B, ...$

Cualquier Estructura-AP es un retículo de subconjuntos cuyos extremos superiores son sus conjuntos de expansión. Alternativamente, podemos definir una Estructura-AP como el retículo definido por 1 o varios Conjuntos-AP entre los cuales no existe la propiedad de inclusión.

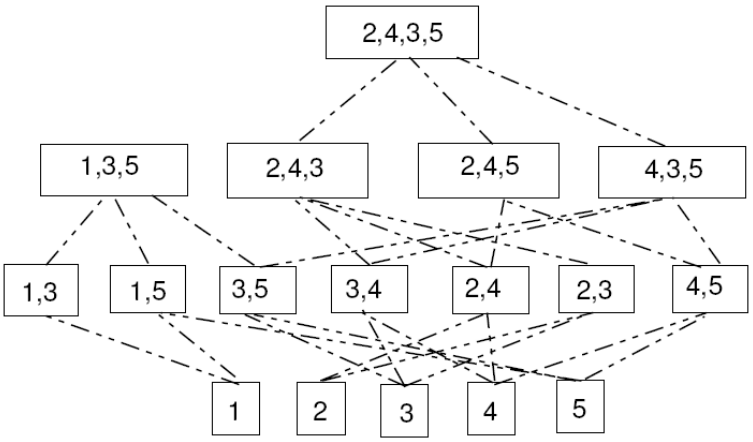


Figura 2.2: Retículo de una Estructura-AP

Las Estructuras-AP tienen las siguientes propiedades:

Inclusión de Estructuras-AP

Definición Sean $\mathcal{T}_1$ y $\mathcal{T}_2$ dos Estructuras-AP con el mismo referencial:
$\mathcal{T}_1 \subseteq \mathcal{T}_2 \Leftrightarrow \forall R$ *Conjunto-AP de* $\mathcal{T}_1$,
$\exists \mathcal{S}$ *Conjunto-AP of* $\mathcal{T}_2$ *tal que* $\mathcal{R} \subseteq \mathcal{S}$

Una Estructura-AP $\mathcal{T}_1$ está incluida en otra $\mathcal{T}_2$ si sólo si, para cada uno de los Conjunto-AP $\mathcal{R}$ en $\mathcal{T}_1$ existe un Conjunto-AP $\mathcal{S}$ en $\mathcal{T}_2$ que lo incluye.

Subestructura-AP inducida

Definición Sea una Estructura-AP $\mathcal{T} = g(A_1, A_2, ..., A_n)$ con referencial X y $Y \subseteq X$ se define una Subestructura-AP de $\mathcal{T}$ inducida por Y:
$\mathcal{T}' = \mathcal{T} \bigwedge Y = g(B_1, B_2, ..., B_m)$
donde
$\forall B_i \in \{B_1, ..., B_m\} \Rightarrow \exists A_j \in \{A_1, A_2, ..., A_n\}$ *tal que* $B_i = A_j \bigcap Y$
$\forall A_j \in \{A_1, ..., A_n\} \Rightarrow \exists B_i \in \{B_1, B_2, ..., B_m\}$ *tal que* $A_j \bigcap Y \subseteq B_i$

$\mathcal{T}'$ es la Estructura-AP generada por la intersección de los conjuntos de expansión de $\mathcal{T}$ e Y, sin considerar las intersecciones no minimales.

Superestructura-AP inducida

Definición Sea una Estructura-AP $\mathcal{T} = g(A_1, A_2, ..., A_n)$ con referencial X y $Y \subseteq X$, se define una Superestructura-AP de $\mathcal{T}$ inducida por Y:
$\mathcal{T}' = \mathcal{T} \bigvee Y = g(B_1, B_2, ..., B_m)$
donde
$\forall B_i \in \{B_1, ..., B_m\} \Rightarrow \exists A_j \in \{A_1, A_2, ..., A_n\}$ *tal que* $B_i = A_j \bigcup Y$
$\forall A_j \in \{A_1, ..., A_n\} \Rightarrow \exists B_i \in \{B_1, B_2, ..., B_m\}$ *tal que* $A_j \bigcup Y \subseteq B_i$

$\mathcal{T}'$ es la Estructura-AP generada por la unión de los conjuntos de expansión de $\mathcal{T}$ e Y, sin considerar las uniones que generan Conjuntos-AP no minimales.

De lo anterior se deduce la siguiente propiedad:

Definición Sea $\mathcal{T} = g(A_1, A_2, ..., A_n)$, Y_1 y Y_2 Estructuras-AP y subconjuntos con el mismo referencial, entonces
$((T \bigwedge Y_1) \bigwedge Y_2) = T \bigwedge (Y_1 \bigcap Y_2)$
$((T \bigvee Y_1) \bigvee Y_2) = T \bigvee (Y_1 \bigcup Y_2)$

La inducción iterativa de Subestructuras-AP sobre conjuntos $\{Y_1, Y_2, ..., Y_n\}$ es equivalente a la inducción de la Subestructura-AP sobre la intersección de dichos conjuntos. De forma análoga, la inducción iterativa de Superestructuras-AP sobre conjuntos $\{Y_1, Y_2, ..., Y_n\}$ es equivalente a la inducción de la Subestructura-AP sobre la unión de dichos conjuntos.
Las operaciones $\bigwedge$ y $\bigvee$ se definen sobre conjuntos y Estructuras-AP. Sobre las operaciones que se pueden realizar entre Estructuras-AP, son las siguientes:

Unión de Estructuras-AP:

Definición Sea $\mathcal{T}_1 = g(A_1, A_2, ..., A_n)$, $\mathcal{T}_2 = g(B_1, B_2, ..., B_n)$ dos Estructuras-AP, se define
$\mathcal{S} = \mathcal{T}_1 \bigcup \mathcal{T}_2 = g(C_1, ..., C_h)$
verificando

$\forall C_i \in \{C_1, ..., C_h\}$; $(\exists A_j / C_i = A_j)$ or $(\exists B_l / C_i = B_l)$
$\forall A_j \in \{A_1, ..., A_n\}$; $\exists C_i / A_i \subseteq C_i$
$\forall B_l \in \{B_1, ..., B_m\}$; $\exists C_i / B_l \subseteq C_i$

La unión de Estructuras-AP $\mathcal{T}_1 \bigcup \mathcal{T}_2$ consiste en fusionar los correspondientes conjuntos de expansión evitando valores redundantes.

Intersección de Estructuras-AP:

Definición Sea $\mathcal{T}_1 = g(A_1, A_2, ..., A_n)$, $\mathcal{T}_2 = g(B_1, B_2, ..., B_n)$ dos Estructuras-AP, se define
$\mathcal{S} = \mathcal{T}_1 \bigcap \mathcal{T}_2 = g(C_1, ..., C_h)$
verificando
$\forall C_i \in \{C_1, ..., C_l\}$
$\forall A_p \in \{A_1, ..., A_n\}$,
$\{B_q \in B_1, ..., B_m\}$; $C_i = A_p \bigcap B_q$

La intersección de Estructuras-AP $\mathcal{T}_1 \bigcap \mathcal{T}_2$ consiste en hacer la intersección entre los correspondientes conjuntos de expansión de $\mathcal{T}_1$ y $\mathcal{T}_2$ evitando valores redundantes.

TDA Estructura-AP

Una vez definidas formalmente las estructuras, se profundiza en la definición y creación de un tipo de dato abstracto (TDA) que permita manipular dichas estructuras. El objetivo perseguido es representar los campos textuales no estructurados como objetos del TDA definido, permitiendo así dotarlos de estructura manteniendo el significado hasta cierto punto. Además el TDA permite realizar las operaciones vistas anteriormente.

A partir de los datos originales el artículo propone una metodología para la creación de los Conjuntos-AP y los TDA. Las tareas a realizar son las siguientes:

1. Obtener un diccionario de datos compuesto de una lista de términos eliminado las stop-words. Adicionalmente es conveniente sustituir ciertos términos por formas equivalentes como ocurre con sinónimos, acrónimos, etc... para evitar que la diversidad de formas sintácticas que se refieren a un mismo término hagan bajar la frecuencia de este.

2. Transformar los datos en una base transaccional donde los términos del diccionario de datos son los atributos y cada tupla se corresponde con un registro. El valor de un atributo para una determinada tupla será 1 si dicho término aparece en el registro original y 0 en el caso contrario.

3. Se obtienen los itemsets frecuentes de la base de datos transaccional anterior utilizando el algoritmo Apriori [Agrawal and R.Srikant, 1994].

4. Los itemsets maximales (Conjuntos-AP) forman la estructura del retículo según la propiedad Apriori

5. Todos los Conjuntos-AP forman una Estructura-AP que se representará en el TDA.

La Estructura-AP obtenida no cubre toda la información contenida en el campo textual, pero se considera como el dominio activo del atributo, denotado $\mathcal{D}$.

Sea A un atributo de la base de datos que contiene un texto breve, por cada tupla t de la base de datos, Y_t es el conjunto de términos obtenidos de $t[A]$ tras realizar el preprocesamiento. La Estructura-AP correspondiente al valor A de t viene dada por:

$\mathcal{T} = Y_t \bigwedge \mathcal{D}$

Dado que la operación de inducción de Subestructuras-AP está incluida en el TDA podemos representar cada texto breve como una Estructura-AP, únicamente aplicando la fórmula anterior. Esto permitiría ahorrar espacio de almacenamiento puesto que el valor no tendría que almacenarse explícitamente, pero en cambio el costo de procesamiento aumentaría.

Consulta de Estructuras-AP

Cuando los Conjuntos-AP están expresados como TDAs pueden ser consultados y procesados. Para realizar una consulta, los usuarios expresan sus necesidades de información como un conjunto de términos. En la base de datos, los valores de los atributos se representan mediante Estructuras-AP. Dependiendo de la correspondencia entre los términos de la consulta y la Estructura-AP se recuperarán o no los registros correspondientes. Se definen dos posibles correspondencias o acoplamientos.

Correspondencia Fuerte

Definición Sea una Estructura-AP $\mathcal{T} = g(A_1, A_2, ..., A_n)$ con referencial X y $Y \subseteq X$ se define la correspondencia fuerte entre Y y $\mathcal{T}$ como una operación lógica.

$$Y \bigodot T = \begin{cases} verdadero\ si & \exists A_i \in \{A_1, A_2, ..., A_n\} / Y \subseteq A_i \\ falso & \text{en otro caso} \end{cases}$$

En este caso se dice que la correspondencia entre un conjunto Y y una Estructura-AP $\mathcal{T}$ es fuerte cuando el conjunto consulta Y es subconjunto de alguno de los conjuntos de expansión A_i de la Estructura-AP.

Correspondencia Débil

Definición Sea una Estructura-AP $\mathcal{T} = g(A_1, A_2, ..., A_n)$ con referencial X y $Y \subseteq X$ se define la correspondencia débil entre Y y $\mathcal{T}$ como una operación lógica.

$$Y \bigoplus T = \begin{cases} verdadero\ si & \exists A_i \in \{A_1, A_2, ..., A_n\} / Y \bigcap A_i \neq \emptyset \\ falso & \text{en otro caso} \end{cases}$$

La correspondencia entre un conjunto Y y una Estructura-AP $\mathcal{T}$ se dice débil si la intersección del conjunto consulta Y y alguno de los conjuntos de expansión A_i de la Estructura-AP, es no vacía.

Estas medidas pueden cuantificarse empleando medidas o índices. El valor del índice vendrá dado por el número de términos que tienen correspondencia y el número de conjuntos de expansión con los que se corresponden. Cuantos más términos tengan correspondencia y la tengan con más conjuntos de expansión, mayor será el índice. Se define un índice para cada una de las medidas vistas anteriormente.

Índices de correspondencia fuerte/débil

Definición Sea $\mathcal{T} = g(A_1, A_2, ..., A_n)$ una Estructura-AP con referencial X y sea $Y \subseteq X$. Se define el índice de correspondencia fuerte (análogo débil) entre Y y $\mathcal{T}$ como:
$\forall A_i \in \{A_1, A_2, ..., A_n\}$ se denota $m_i(Y) = card(Y \bigcap A_i)/card(A_i)$, $S = \{i \in \{1, ..., n\} | Y \subseteq A_i\}$, $W = \{i \in 1, ..., n\} | Y \bigcap A_i \neq \emptyset\}$

$m_i(Y)$ mide la cantidad de términos que verifican la correspondencia y toma valores en el rango [0,1].

Por tanto, se definen los índices de correspondencia fuerte y débil entre Y y $\mathcal{T}$ como:

Índice de Correspondencia Fuerte $= S(Y|\mathcal{T}) = \sum_{i \in S} m_i(Y)/n$

Índice de Correspondencia Débil $= W(Y|\mathcal{T}) = \sum_{i \in W} m_i(Y)/n$

Por tanto:
$\forall Y$ y $\mathcal{T}$, $S(Y|\mathcal{T}) \in [0, 1]$, $W(Y|\mathcal{T}) \in [0, 1]$ y $W(Y|\mathcal{T}) \geq S(Y|\mathcal{T})$

La medida del índice es la media de los valores $m_i(Y)$ para los conjuntos que verifican la correspondencia para cada una de las medidas. Los rangos de valores de índice están en el intervalo [0,1] y para un mismo conjunto y Estructura-AP, el valor del índice para la correspondencia débil será siempre menor o igual que para la fuerte.

La consulta de la base de datos puede realizarse desde diferentes puntos de vista.

1. Se puede consultar toda la información incluida en la base de datos para un determinado atributo, esto es, consultar el dominio activo $\mathcal{D}$

2. Se proporciona una lista inicial de términos de consulta sin conocer el vocabulario de la Estructura-AP. A través del Dominio activo $\mathcal{D}$ se comprueba si los términos son adecuados y en caso de no serlo se sugieren términos más adecuados.

3. Se proporciona un conjunto de términos a modo de consulta, se comprueba la correspondencia entre los términos y las Estructuras-AP en los atributos, obteniendo índices de correspondencia fuerte y débil. Es posible definir umbrales para evitar recuperar información poco relevante.

Implementación en una Base de Datos Objeto-Relacional

Para poder utilizar la representación de Conjuntos-AP y Estructuras-AP es necesario incluir ciertos metadatos.

Una *relación* está compuesta de *atributos*, en el caso de que estos atributos sean textos breves, estos tienen un dominio activo. El dominio activo puede representarse como un texto breve o como una Estructura-AP, en el primer caso, éste puede representarse como un conjunto de términos, y dichos términos pueden verse como una Estructura-AP.

Cuando se tiene un atributo de tipo texto breve, éste se representa como un conjunto de términos y una Estructura-AP para cada una de las tuplas de la relación, y además debe tenerse una Estructura-AP que represente el dominio activo para todos los valores del atributo.

Para poder realizar esta doble representación, se utiliza el campo original para almacenar la Estructura-AP correspondiente y se añade a la relación un campo adicional donde se almacenan los conjuntos de términos.

Para representar el dominio activo de un atributo, se crea una tabla de catálogo `AP-Description` donde se indica el valor de la Estructura-AP que representa el dominio activo junto con los identificadores de la relación y atributo al que pertenece.

En cuanto a la representación de Conjuntos-AP y Estructuras-AP propiamente dichas, éstas son grafos dirigidos por lo que existen dos alternativas.

1. Representar únicamente los conjuntos de expansión
   Esta alternativa es sencilla de implementar y el espacio de almacenamiento necesario no es excesivo.

2. Representar los grafos completos
   Se requiere mucho más espacio de almacenamiento pero permite representar información adicional, como por ejemplo el soporte de cada Conjunto-AP. Una posibilidad para reducir el espacio de almacenamiento es no representar todos los nodos, sólo las variaciones de un nodo respecto al anterior.

En cuanto a la consulta, será necesario implementar métodos de consulta sobre los metadatos y sobre los datos convencionales. En el primer caso se pueden formular consultas para recuperar los conjuntos de expansión de una Estructura-AP, obtener subconjuntos de una Estructura-AP, etc... En el caso de la consulta de datos convencionales se aplican los métodos de correspondencia fuerte y débil.

2.2.3. Aprendizaje de Ontologías utilizando WordNet

Una vez hemos visto con detalle el proceso de aprendizaje de ontologías a partir de textos, en este apartado nos centraremos en la utilización de distintas bases de conocimiento que sean de utilidad en este proceso. Concretamente en este apartado nos centraremos en el uso de WordNet.

En [Cimiano and Staab, 2005] Se presenta una aproximación para la inducción de jerarquías conceptuales a partir de colecciones de texto.

El cálculo de la similitud entre términos se realiza teniendo en cuenta la hipótesis distribucional de Harris [Harris, 1968] según la cual, dos términos son semánticamente similares en la medida en la que comparten contextos sintácticos similares.

Por cada término se obtienen dependencias sintácticas superficiales sobre etiquetas Part-of-speech.

Se construye un oráculo de hiperónimos, una función que para un término dado devuelve un conjunto de tuplas que indican en posible hiperónimo, y un valor numérico indicando el número de veces que el algoritmo ha encontrado evidencia de la relación de hiperonimia entre los términos. Los hiperónimos se buscan en tres fuentes, WordNet, patrones de Hearst [Hearst, 1992] sobre un corpus y patrones de Hearst sobre la Web.

2.3. Medidas de Relación Semántica

En este apartado vamos a comentar algunas técnicas complejas para determinar la relación semántica entre conceptos.

Es importante distinguir entre relación semántica y similitud semántica, puesto que existen importantes diferencias entre ambos conceptos. Tomemos como ejemplo las palabras *mesa* y *mantel*, estas palabras están altamente relacionadas, puesto que un mantel es un trozo de material que se coloca habitualmente sobre una mesa, para evitar que ésta se ensucie. Sin embargo, la similitud entre ambos términos es baja, puesto que ni tienen una forma similar, ni están hechos del mismo material, etc... De este modo la similitud entre ambos términos sería baja, mientras que la relación sería bastante alta.

Teniendo esto en cuenta, podemos ver que las medidas de similitud semántica son más específicas y responden a criterios de valoración basados en relaciones jerárquicas del tipo *IS-A* (hiperonimia, hiponimia). Por tanto se indica especialmente el uso de estas medidas en jerarquías o taxonomías conceptuales.

Por otra parte, el concepto de relación semántica es más amplio y a su vez incluye al concepto de similitud semántica. En este caso las relaciones tenidas en cuenta no sólo son de tipo jerárquico sino que se tienen en cuenta relaciones *es parte de*, *antónimo de*, *relacionado con*, *es atributo de*, etc...

2.3.1. Medidas de Relación Semántica empleando WordNet

Banerjee-Pedersen (text-overlap)

Esta medida de solapamiento de texto [Banerjee and Pedersen, 2003], es una medida de relación semántica, en cuanto indica si dos términos están relacionados, y emplea como base de conocimiento externa WordNet.

Esta medida determina la relación semántica entre dos términos, que se basa en la obtención de información a partir de diccionarios automáticos. Particularmente se diseñó para aprovechar la estructura de jerarquías o taxonomías de conceptos como los presentes en recursos como WordNet. En este tipo de recursos cada concepto tiene asociado un significado, y existe una pequeña descripción textual que define dicho significado (*gloss*). Para determinar la relación entre un par de conceptos se calcula el solapamiento entre sus

descripciones y las descripciones relacionadas con ellos (hiperónimos, hipónimos, sinónimos, etc...). El solapamiento (*overlap*) entre dos textos se define como el conjunto de palabras que tienen en común.

A la hora de calcular el solapamiento entre dos definiciones, se tiene en cuenta la longitud de la secuencia de palabras consecutivas que aparecen en ambas definiciones. Este proceso ignora las palabras funcionales (pronombre, preposiciones, artículos, conjunciones) cuando se encuentran a principio o final de la frase, pero se contabilizan cuando se encuentran en posiciones intermedias de la secuencia. Cada vez que se detecta un solapamiento éste se elimina del texto original y es sustituido por un marcador, posteriormente se vuelve a realizar el análisis en busca de solapamientos. El proceso continúa hasta que no es posible encontrar más solapamientos. El valor obtenido de solapamiento (*score*) se calcula sumando la longitud al cuadrado de cada solapamiento, esto se hace para dar más relevancia a solapamientos de mayor longitud, frente a solapamientos cortos o de palabras individuales.

La medida se expresa de la siguiente forma:

$$\begin{gathered} relatedness(A,B) = \sum score(R_1(A), R_2(B)), \\ \forall (R_1, R_2) \in RELPAIRS, \end{gathered} \tag{2.1}$$

donde R_1 son funciones R_2 que devuelven la definición para el tipo de relación (los tipos de relación pueden ser hipónimos, hipernónimos, etc...), $score(x, y)$ es una función que devuelve el cálculo del solapamiento entre dos textos y $RELPAIRS$ contiene los conjuntos de pares de relaciones a incluir en el cálculo, la única restricción es que para cada par de relaciones incluidas, debe de incluirse también su inversa, para garantizar la reflexividad de la medida.

2.3.2. Medidas de Relación Semántica empleando Wikipedia

En este apartado analizaremos brevemente algunos de los trabajos que emplean Wikipedia para obtener medidas de relación semántica entre términos.

WikiRelate!

En [Strube and Ponzetto, 2006] se utiliza por primera vez Wikipedia para el cálculo de medidas de relación semántica. Esta propuesta toma técnicas empleadas habitualmente en WordNet y las adapta para el uso con Wikipedia.

El proceso seguido por este método consiste en, dado un par de palabras i y j, se recuperan las páginas a las que hacen referencia los términos y se realiza una desambiguación, posteriormente se realiza un proceso de búsqueda en el árbol de categorías y finalmente se realiza el cálculo de la medida de relación semántica.

Detallaremos cada uno de los diferentes pasos:

- *Recuperación de páginas y desambiguación*: Para recuperar las páginas p_i y p_j a las que hacen referencia los términos i y j, primero se busca por el título de las páginas. A continuación se siguen las páginas de redirección para resolver ambigüedades sobre las páginas obtenidas en la búsqueda. La filosofía seguida por este método es la de

maximizar la relación semántica entre páginas, por lo que primero se intenta que las consultas de búsqueda sobre las páginas se desambiguen la una a la otra. Si al consultar por un término i se alcanza una página de desambiguación p_i, se obtienen todos los enlaces de la página p_j obtenida al consultar por el término j *sin desambiguar*. Si un enlace en p_i contiene alguna ocurrencia del término j se devuelve la página enlazada. Si no se encuentra el término j en los enlaces de la página p_i, se devuelve la página correspondiente al primer enlace en la página de desambiguación. En el caso de que ambas páginas sean ambiguas, se toman los términos desambiaguados de p_j como una lista de desambiguación y se repite el proceso anterior para cada término. De forma análoga el proceso se repite para p_j con el término i, de modo que las consultas se desambiguan entre sí atendiendo a la relación existente entre ellas. Aunque la estrategia de desambiguación no es muy precisa, ofrece una solución práctica, ya que seguir todos los enlaces de las páginas de desambiguación puede ser costoso debido al elevado número de enlaces presentes en algunas de ellas.

- *Búsqueda en el árbol de categorías*: El texto presente en las páginas de Wikipedia es más que suficiente para poder realizar los cálculos con las medidas de solapamiento. Pero es necesario obtener los caminos en el árbol de categorías, para el cálculo de medidas de distancia y de contenido de la información. Dadas las páginas p_i y p_j, se extraen las categorías C_i y C_j a las que pertenecen. A partir de las listas de categorías, por cada par de categorías $\langle c_i, c_j \rangle, c_i \in C_i, c_j \in C_j$, se realiza una búsqueda de un ancestro común a una profundidad limitada máxima de 4. Este valor viene dado empíricamente por los experimentos realizados en el artículo correspondiente.

- *Cálculo de la medida de relación*: Dado el conjunto de caminos entre pares de categorías, se seleccionan aquellos que satisfagan la definición de las medidas, esto es, el camino más corto para medidas basadas en longitud del camino y el camino que maximiza el contenido de la información para medidas basadas en IC.

Las medidas adaptadas siguiendo este procedimiento son Rada, Wu-Palmer, Leacock-Chodorow, Resnik y Lesk.

Análisis de Semántica Explícita

En [Gabrilovich and Markovitch, 2007] se presenta el análisis de semántica explícita (*Explicit Semantic Analysis*, *ESA*). Este método representa el significado de los textos en un espacio de alta dimensionalidad derivado de Wikipedia. Mediante el uso de técnicas de aprendizaje automático se representa el significado de cualquier texto como un vector ponderado de conceptos extraídos de Wikipedia. Para determinar la similitud entre textos en este espacio, basta con emplear métricas convencionales como la medida del coseno.

La denominación de este método viene dada en contraposición al método de análisis de la semántica latente (*Latent Semantic Analysis*, *LSA*), en el que por medio de técnicas estadísticas se determinan un conjunto de conceptos latentes extraídos del texto. Uno de los problemas de LSA es que los conceptos no siempre son fácilmente interpretables por un humano. En contraposición, el análisis de semántica explícita utiliza conceptos creados

por humanos (los conceptos de Wikipedia) para representar los textos, por lo tanto la interpretación es más intuitiva.

El método tiene una primera parte de preprocesamiento de la información de Wikipedia. Se eliminan todas las marcas HTML y se procesan todas las páginas. Una vez el texto está limpio, cada concepto (página de Wikipedia) se representa como un vector de palabras que aparecen en el artículo. A las entradas del vector se les asigna pesos según el esquema TFIDF [Salton and McGill, 1983], estableciendo la relación entre los términos y el concepto al que están asociados. Para mejorar el rendimiento se crea un índice invertido donde por cada palabra que aparece en Wikipedia, se tiene una lista ponderada de los conceptos en los que aparece dicha palabra.

Este método permite calcular la relación entre textos de tamaño variable, así el caso base en el que se determina la relación entre dos términos textuales, se puede extender a uno más genérico en el que se determina la relación entre textos de mayor longitud.

El método implementa un intérprete semántico que obtiene por cada palabra del texto de entrada, la lista de conceptos en los que aparece, y mediante un clasificador basado en centroides, se ordenan todos los conceptos según su relevancia para el fragmento de texto de entrada. Dados dos textos, el intérprete semántico devuelve dos vectores ponderados, uno por cada texto, cuyos pesos indican la relevancia de cada concepto de Wikipedia para el texto original. Utilizando la medida del coseno [Salton and McGill, 1983] entre ambos vectores se obtiene un valor de relación semántica entre los textos originales.

Atendiendo a la correlación obtenida con respecto a estimaciones realizadas por humanos, este método es de los más efectivos para el cálculo de la relación semántica entre un par de palabras o un par de textos. Sin embargo, esta aproximación tiene la desventaja de la gran carga computacional necesaria para procesar todos los datos, ya que no debemos olvidar que cada concepto es una página de Wikipedia y que la versión en inglés ronda los 3 millones de artículos. Aunque se eliminen muchos artículos por considerarse redundantes o poco apropiados, el volumen de datos con el que trabajar es abrumador y difícilmente será posible realizar un procesamiento en línea para el cálculo de la relación semántica entre textos de un tamaño medio. Pero en la parte positiva está la ausencia de una limitación en el tamaño de los textos, teóricamente los textos a tratar podrían ser de cualquier longitud.

Cálculo de Relación Semántica Empleando la Estructura de los Enlaces de Wikipedia

Esta técnica definida en [Milne and Witten, 2008] recibe el nombre de *Wikipedia Link Vector Model* (WLVM) y calcula la relación semántica entre términos utilizando la estructura de enlaces y los títulos de las páginas de Wikipedia, en lugar de utilizar todo el contenido textual.

El proceso comienza asociando a un término una página, esto se hace buscando en los títulos de los artículos, lo cual a veces es una fuente de ambigüedad. Esta aproximación calcula la relación semántica entre dos artículos de Wikipedia como el ángulo entre los vectores de enlaces encontrados en los artículos. La técnica es similar a la del *Vector Space Model* utilizada en recuperación de información, pero en lugar de construir vectores con frecuencias de términos, se crean vectores con conteos de enlaces ponderados por la

probabilidad de aparición del enlace. Dicha probabilidad se define como el total de enlaces de un artículo fuente a un artículo objetivo sobre el total de enlaces de cualquier artículo hacia el objetivo. Si t es el número total de artículos en Wikipedia, entonces el peso w para un enlace $a \rightarrow b$ se define como:

$$w(a \rightarrow b) = |a \rightarrow b| \cdot log \left(\sum_{x=1}^{t} \frac{t}{|x \rightarrow b|} \right). \tag{2.2}$$

De este modo los enlaces de una página a otra se consideran menos importantes para determinar la similitud entre artículos, si hay muchos otros artículos que también enlazan a esa misma página. El vector de pesos para cada término se rellena con los pesos de cada uno de los enlaces para la página. Una vez obtenidos los vectores de pesos para los enlaces de cada página, se aplica la medida del coseno sobre ambos vectores, el valor obtenido es el valor de relación semántica entre los términos.

JWPL

En [Torsten Zesch, 2007] se estudia la viabilidad del uso del grafo de páginas y el grafo de categorías de Wikipedia como red semántica que puede usarse en tareas de procesamiento de lenguaje natural. Para ello, se adaptan las medidas de similitud y relación semántica desarrolladas para WordNet y se adaptan a Wikipedia. Posteriormente en la línea del anterior trabajo aparece [Zesch and Gurevych, 2010] donde se amplia el estudio acerca del cálculo de medidas de relación semántica entre términos, empleando para ello diversos recursos sobre diferentes conjuntos de datos.

El cálculo de la relación semántica entre dos términos se realiza obteniendo las páginas asociadas a los términos, de entre todas esas páginas se toman pares, cada página perteneciendo a uno de los términos iniciales y se calculan los valores de relación semántica entre páginas, luego se selecciona la media o el máximo de los valores obtenidos para todos los pares. La relación semántica entre dos páginas se calcula como la relación semántica entre las categorías a las que pertenecen dichas páginas. Se tomarían pares de categorías y se aplicarían las medidas de relación semántica adaptadas a Wikipedia. Una vez realizados todos los cálculos se seleccionaría el máximo o la media de los valores obtenidos.

El principal inconveniente que vemos en esta aproximación, es que el grafo de Wikipedia contiene una gran cantidad de categorías de gestión. Categorías que como tales no aportan semántica desde un punto de vista general sino desde un punto de vista concreto de la gestión de los contenidos en Wikipedia. Estas categorías de gestión se emplean en los cálculos de relación semántica y eso desvirtúa en cierta forma las medidas.

Una de las mayores aportaciones de este trabajo es la API para el acceso a Wikipedia JWPL. A partir de los ficheros de copia de seguridad de Wikipedia que se realizan periódicamente, mediante esta herramienta es posible generar ficheros de datos para cargar en una base de datos MySQL. Una vez los datos están almacenados JWPL proporciona métodos de acceso a las páginas, categorías y metadatos almacenados. La herramienta no sólo proporciona acceso a los datos almacenados, sino que proporciona utilidades para la evaluación de medidas de similitud semántica sobre Wikipidea. Para ello proporciona

mecanismos para la recreación del grafo de categorías de Wikipedia, su adaptación a una estructura en árbol y el cálculo de estructuras necesarias para la obtención de resultados con las medidas de relación semántica.

2.3.3. Consultas Conceptuales

La consulta conceptual consiste en la abstracción conceptual de un conjunto de documentos o textos de una base de datos. En [Andreasen and Bulskov, 2009] la técnica se aplica para permitir la visualización de forma rápida para grandes volúmenes de datos. El objetivo es restringir una ontología general que contenga conocimiento diverso, al conjunto de conceptos que aparecen en los textos analizados. La exploración conceptual del conjunto de documentos se puede realizar extrayendo los conceptos del texto y proporcionando métodos para la navegación y recuperación de los documentos originales. La ontología de conocimiento general a la que se aplica es WordNet.

2.4. Conclusiones

En este capítulo hemos realizado repaso breve y muy general a algunas de las propuestas relacionadas con el trabajo que vamos a desarrollar. El objetivo es el de poder contextualizar y definir de forma general los trabajos realizados en las diversas áreas en las que se enmarcan nuestras propuestas. Los trabajos directamente relacionados con los desarrollos que vamos a realizar serán tratados con mas detalle en la misma sección donde se realizan las propuestas, para poder explicar mejor las nuevas aproximaciones que se proponen.

Hemos visto como se está comenzando a aceptar la idea de usar ontologías de dominio para el diseño de esquemas de bases de datos. Aunque existen múltiples propuestas, algunas que realizan la transformación mediante reglas, mediante algoritmos, etc... no parece que actualmente haya ninguna que destaque sobre las demás. Quizás el principal problema es que aún no se ha asumido completamente el rol que desempeñan las ontologías como estructuras para la representación del conocimiento, y se pretende diseñar ontologías válidas para múltiples escenarios, cuando se debería pensar más en el uso de ontologías más específicas de uso concreto y acotado, pero que se pudiesen relacionar o conectar con el resto para desarrollar acciones más complejas.

Por lo que respecta a la representación de la semántica en bases de datos es un problema de suma importancia que ha sido abordado desde múltiples puntos de vista. Desde la perspectiva del almacenamiento de ontologías, hemos visto como a pesar de la gran cantidad de propuestas que se han realizado, la mayoría de ellas han caído en desuso y pocas han alcanzado el estatus de producto comercial como Openlink Virtuoso. Actualmente el sistema de almacenamiento de ontologías más maduro y cuyo uso está mas extendido es Sesame.

El aprendizaje ontológico ha sido un campo muy activo en los últimos años, pero en este caso parece que el problema vuelve a repetirse, quizás no tenga sentido aprender una gran ontología que contenga todo el conocimiento posible. El objetivo debería ser aprender ontologías específicas para desarrollar una tarea concreta.

Finalmente resaltar la gran relevancia que están adquiriendo en los últimos años WordNet y Wikipedia como herramientas de apoyo a tareas de procesado del lenguaje y para la representación semántica.

Capítulo 3

Generación de Esquemas de Bases de Datos a partir de Ontologías OWL

El uso cada vez mas extendido de las ontologías como formalismos para la representación del conocimiento, y particularmente su aplicación en el entorno de la Web Semántica, ha motivado la aparición de multitud de ontologías escritas en OWL que contienen grandes cantidades de datos sobre instancias. De forma paralela está tomando gran relevancia un movimiento conocido como Linked Data que trata de definir métodos para publicar, compartir y conectar datos de la Web Semántica a través de URIs y RDF. Esta última aproximación se aleja un poco del uso de una ontología general que defina las relaciones entre distintos datos y se centra más en establecer enlaces entre las diferentes instancias disponibles.

La cantidad de conjuntos de datos RDF (instancias de ontologías) a disposición pública es realmente grande. Los sistemas gestores de datos para la web semántica siguen siendo aún poco apropiados para manejar datos que pueden llegar a los cientos de millones de triples. Un ejemplo claro de este tipo de datos y su relevancia puede encontrarse en el conjunto de datos de DBPedia [Auer et al., 2008] (instancias RDF obtenidas a partir de Wikipedia [Wikipedia, 2010]), más concretamente en el conjunto de datos de Infoboxes.

Los Infoboxes son fragmentos de información estructurada que se insertan en las páginas wiki, para mostrar datos relevantes. De este modo el infobox de una página wiki dedicada a una ciudad contendrá su nombre, el país al que pertenece, la región y coordenadas geográficas, etc. Dado que estos datos contienen una estructura subyacente es posible extraerlos de forma semiautómatica. Actualmente se realiza un proceso en el que a partir de la estructura de los infoboxes, se efectúan de forma manual una serie de mapeos a una ontología definida para estos. Posteriormente se extraen de Wikipedia los datos de instancia de los infoboxes atendiendo a esta definición. La ontología definida se denomina ontología de los Infoboxes, y junto a ella se ponen a libre disposición una elevada cantidad de instancias extraídas automáticamente de los infoboxes de Wikipedia. En la última versión disponible de DBpedia, la ontología de los infoboxes tiene alrededor de 200 clases definidas y 500 propiedades, un tamaño mediano, pero no especialmente grande si se compara con otras ontologías más complejas. Sin embargo, el conjunto de instancias para esa ontología consta

de 9.8 millones de triples. Lo cual es una cantidad bastante considerable. Cada vez más nos encontramos en esta situación en la que la gestión de la ontología no supone un gran esfuerzo, pero la gran cantidad de instancias disponible nos empuja a buscar soluciones de gestión para grandes volúmenes de datos.

Las bases de datos relacionales y sus variantes objeto-relacionales han demostrado con creces su capacidad para operar con grandes volúmenes de datos, es por esto que nuestra intención es la de establecer un modelo en dos capas en el que la capa conceptual esté definida por una ontología (ya que han demostrado ser herramientas muy eficaces para este tipo de tareas), mientras que la parte de gestión de datos será delegada en un SGBD objeto-relacional que pueda resolver de forma eficiente las consultas sobre un gran volumen de datos de instancias.

En este contexto, nuestra propuesta consiste en el uso de ontologías de dominio como modelos conceptuales para diseñar el esquema de bases de datos en el que se van a almacenar las instancias. La tarea de diseño de un esquema de base de datos no es trivial, ya que requiere conocimientos precisos acerca de dominio del problema a modelar, y conocimiento acerca del modelo conceptual a usar. El diseño de ontologías de dominio tampoco es una tarea trivial, pero es obligatoria a la hora de desarrollar sistemas para la Web Semántica. Una vez definida una ontología en OWL no es necesario crear otro modelo conceptual para el diseño de la base de datos (modelo Entidad-Relación). La ontología contiene todo el conocimiento necesario para desarrollar un esquema apropiado para los datos.

En nuestra opinión es importante preservar la definición de la ontología explícitamente, ya sea de forma externa o dentro de la propia base de datos, con el objetivo de usarla para tareas de concordancia semántica. También es importante mantener las instancias de la ontología estructuradas y separadas de la ontología, para facilitar el acceso a ellas. Teniendo en cuenta todas estas consideraciones, proponemos un esquema híbrido para el almacenamiento de la ontología y de sus instancias, de modo que sea posible utilizarlas de forma conjunta o por separado.

La transformación de ontología de dominio OWL DL a esquema relacional se realiza a través de un algoritmo. Este algoritmo toma como entrada un fichero OWL con la definición de la ontología, y genera un esquema relacional de bases de datos para almacenar las instancias. La ontología se almacena en tablas de catálogo especialmente diseñadas para tal efecto (estructura $TBox$), y las instancias (aserciones de la $ABox$) se insertan en el esquema creado previamente, además se establecen relaciones entre los datos del esquema y sus correspondientes conceptos almacenados en la ontología.

En este capítulo se presenta un esquema para el almacenamiento de ontologías y un algoritmo para crear y almacenar datos en un esquema de base de datos, creado a partir de una ontología expresada en OWL DL. En el esquema obtenido se almacenará tanto la ontología como sus instancias, reduciendo el esfuerzo que supone diseñar sistemas de Web Semántica desde cero.

3.1. Relación entre Bases de Conocimiento y Bases de Datos

Antes de comenzar a hablar de la relación existente entre Bases de Datos (Databases, DB) y Bases de Conocimiento (Knowledge Bases, KB), consideramos necesario abordar el problema relativo a la terminología.

En ocasiones se emplean los términos ontología y base de conocimiento como sinónimos, cuando deberían de hacerse ciertas puntualizaciones al respecto. La relación entre ontologías y bases de conocimiento es un tema controvertido. No está del todo claro si una ontología sólo debe contener información sobre el esquema abstracto del dominio, o también puede contener instancias concretas de conceptos abstractos. Si se intenta trazar una línea divisoria entre la definición de conceptos abstractos y las definiciones a nivel de instancia, nos encontramos que en ocasiones los conceptos abstractos están formulados en base a instancias concretas (este fenómeno, por ejemplo, se refleja en la propiedad `owl:hasValue` del lenguaje de definición de ontologías OWL).

Para hacer una primera distinción entre ambos conceptos acudimos a una cita de [Gruber, 1995] donde se indica que "Una ontología compartida sólo necesita describir el vocabulario para expesar un dominio, mientras que una base de conocimiento debe incluir el conocimiento necesario para resolver un problema o responder consultas arbitrarias sobre el dominio".

Algunos autores han tratado de resolver este problema identificando los límites y relaciones entre las diferentes definiciones existentes. En [Guarino and Giaretta, 1995] se realiza un extenso análisis de la terminología desde un punto de vista lingüístico y formal. Atendiendo a la definición de una ontología tal y como se define en [Gruber, 1993] se reformula y precisa esta definición dotándola de dos sentidos, "Una teoría lógica que da una explicación explícita parcial sobre una conceptualización" y "Una estructura semántica intensional que codifica las reglas implícitas que modelan un fragmento de la realidad".

Atendiendo a esta definición reformulada, podemos hacer la distinción entre ontología y base de conocimiento, diciendo que una ontología es un caso particular de base de conocimiento en el que sólo tenemos información acerca de la parte intensional que describe el dominio.

En nuestro caso particular, optaremos en lo sucesivo por utilizar el término base de conocimiento para referirnos al uso conjunto de información de carácter intensional y extensional, y emplearemos el término ontología para hacer referencia a la parte intensional de una base de conocimiento. Particularmente en este capítulo hablaremos de bases de conocimiento basadas en lógica descriptiva.

Una vez hemos establecido un marco de referencia con respecto a la nomenclatura y las diferencias principales entre ontologías y bases de conocimiento, pasaremos a analizar las similitudes existentes entre bases de conocimiento y bases de datos.

Aunque en un primer acercamiento superficial podría parecer que los términos Base de Conocimiento y Base de Datos describen sistemas similares, un análisis más pormenorizado nos hace ver las múltiples diferencias y en ocasiones incompatibilidades que encierran ambos modelos.

Comenzaremos realizando un breve repaso a los métodos de representación del conocimiento en la lógica de descripciones (Description Logics, DL). La creación de sistemas de

conocimiento DL implica dos aspectos principales.

El primero consiste en caracterizar de forma precisa la base de conocimiento. Esta caracterización consiste en determinar de forma apropiada el tipo de conocimiento a manejar por el sistema y la definición clara de los servicios de razonamiento que el sistema debe proporcionar. En este contexto entendemos como razonamiento el tipo de respuestas que el sistema debe proporcionar. El segundo aspecto está relacionado con el desarrollo de mecanismos e interfaces que permitan al usuario un interacción rica y efectiva con los diferentes servicios proporcionados.

En nuestro caso centraremos la atención en el primer aspecto, la definición del sistema. En el caso de DL la descripción funcional de un sistema se puede definir en base a un interfaz conocida como "Tell&Ask". Esta interfaz permite la definición de operaciones que permiten la construcción de la base de conocimiento y operaciones que permiten obtener información de la base de conocimiento mediante consulta.

En DL se establece una clara diferenciación en el tipo de conocimiento que se va a manejar, por una parte está el conocimiento *intensional*, es decir el conocimiento acerca del dominio del problema, y por otro lado tenemos el conocimiento *extensional* específico a cada problema. En una base de conocimiento DL esta distinción se materializa en dos componentes muy bien diferenciadas la $TBox$ y la $ABox$. La $TBox$ contiene el conocimiento sobre la parte intensional del problema, la definición de las propiedades generales y relaciones entre conceptos, en definitiva, la estructura o taxonomía. Por otra parte, la $ABox$ contiene todo el conocimiento específico de los individuos del dominio.

Aparte de la evidente separación conceptual en cuanto al tipo de conocimiento expresado, podemos observar que esta distinción también se mantiene si tenemos en cuenta un criterio de evolución temporal. El conocimiento de la $TBox$ no debería cambiar en el tiempo, puesto que si el problema está bien definido no es usual que cambie la descripción de éste (el conocimiento almacenado sobre el problema). Por otra parte, el conocimiento extensional contenido en la $ABox$ tiene un carácter más temporal y dependiente, pudiendo experimentar cambios.

De forma análoga, en los sistemas de bases de datos la definición del esquema codifica el conocimiento intensional que se tiene acerca del dominio del problema, y por otra parte las instancias almacenadas dentro del esquema componen una suerte de definición extensional de éste. De este modo el esquema de una Base de datos se compara con la $TBox$ de una KB y las intancias con todos los datos serían el equivalente a la $ABox$. Sin embargo la semántica de la $ABox$ difiere de la utilizada para las instancias en la base de datos.

Continuamos con el análisis, analizando el paralelismo existente entre bases de datos y bases de conocimiento basadas en lógica descriptiva.

La relación entre Bases de Datos y Lógica Descriptiva es bastante fuerte. Por lo general, existen sistemas donde los sistemas de representación del conocimiento basados en DL coexisten con un SGBD. En este escenario los SGBDs se encargan de la gestión de grandes cantidades de datos, mientras que los sistemas de representación del conocimiento gestionan el conocimiento intensional, esta gestión se suele realizar directamente en memoria.

La lógica descriptiva proporciona un entorno formal bastante similar al proporcionado por lenguajes de modelado semántico de datos como pueden ser el modelo Entidad-Relación [Calvanese et al., 1998]. Modelando un problema en DL y utilizando sus capacidades de

razonamiento, se puede verificar la corrección de éste, determinando en tiempo de diseño si, por ejemplo, la definición de una clase puede contener instancias (una clase definida como la intersección de otras dos clases que han sido definidas como disjuntas es vacía, y por tanto una clase definida como tal sería errónea puesto que no contendría instancias).

Una de las diferencias más relevantes entre las BD y las KB basadas en DL es que las primeras aplican una hipótesis de mundo cerrado, frente a la aplicación de una semántica de mundo abierto por parte de las segundas. Mientras que una base de datos representa exactamente una interpretación, aquella en la que clases y relaciones en el esquema se interpretan como los objetos y tuplas de las instancias, una *ABox* representa muchas interpretaciones diferentes, todos sus modelos. De este modo la ausencia de información en una base de datos se interpreta como una información negativa (mundo cerrado, "closed-world semantics"), mientras que la ausencia de información en una *ABox* únicamente indica la falta de conocimiento (mundo abierto, "open-world semantics") [Baader and Werner, 2003]. Así pues, mientras se considera que la información en una base de datos es completa, se asume que la información sobre una *ABox* es incompleta.

La resolución de una consulta en una base de datos no es un razonamiento lógico, en realidad no se trata mas que de la prueba de un modelo finito, es decir, la evaluación de una fórmula en un modelo finito fijo. Sin embargo, en una *ABox* la consulta es más compleja, ya que requiere de un proceso de razonamiento no trivial, en el que debemos tener en cuenta todos los posibles (en ocasiones infinitos) modelos.

Por otra parte las ontologías no soportan el concepto de clave primaria, que es básico para entender el modelo relacional. En una ontología pueden existir dos instancias que tengan distinto nombre sin que ello necesariamente implique que sean individuos distintos.

Debido a estas diferencias fundamentales, optaremos por el uso de un modelo de consulta basado en mundo cerrado, donde únicamente se realizarán tareas de razonamiento básico, tales como la subsunción de clases, y trataremos de derivar claves primarias de las propiedades de la ontología, incluyendo identificadores adicionales en aquellos casos que esto no sea posible.

Este análisis nos ha permitido ver como en ambos sistemas existe una clara separación de la información a nivel funcional y estructural. Por otra parte, dada la falta de capacidad para gestionar datos de instancia por parte de las bases de conocimiento y razonadores basados en lógica descriptiva, se nos plantea la posibilidad de realizar un cambio de modelo de representación, en el que preservando ciertas características de una base de conocimiento, podamos representar en una base de datos toda la información y gestionarla de forma eficiente. Para ello, el conocimiento intensional que determina las ontologías a través de su taxonomía y las relaciones entre conceptos, se representarán mediante un esquema híbrido de base de datos. En este esquema tendremos una representación para la parte intensional de la base de conocimiento, y otra para la parte extensional. Ambas partes estarán relacionadas a través de una serie de metadatos utilizándose en cada caso el esquema de almacenamiento más apropiado.

La parte intensional de este esquema se almacenará en una serie de tablas de metadatos diseñadas a tal efecto, éstas contendrán la definición de la ontología. Por otra parte, la gestión de las instancias de la ontología será delegada en un SGBD, dado que puede manejar grandes cantidades de conocimiento de este tipo. A través de unas estructuras de

almacenamiento de metadatos, se establecerán relaciones entre las definiciones intensional y extensional de la base de conocimiento, permitiendo asociar las instancias con los conceptos a los que pertenecen. De forma adicional, el esquema híbrido que se propone permite el acceso a las instancias como una base de datos tradicional y a través de herramientas de gestión de ontologías.

3.2. Modelo formal

En esta sección repasaremos de forma breve las definiciones formales de los principales modelos con los que vamos a trabajar. Primero recordaremos brevemente la definición del modelo relacional, posteriormente repasaremos la definición formal de los esquemas Entidad Relación y terminaremos con una caracterización de las bases de conocimiento DL.

3.2.1. Modelo Relacional

El modelo relacional fue definido por primera vez en [Codd, 1970]. Este modelo, basado en el álgebra relacional, presentaba una nueva alternativa a los modelos clásicos y en red.

Previamente a la descripción formal del modelo relacional, debemos presentar algunos conceptos que nos serán de utilidad en las definiciones empleadas.

- *Dominio o Tipo* : Es el conjunto de posibles valores que puede adoptar un atributo. El dominio o tipo, puede ser atómico (no descomponible semánticamente), o complejo.
- *Atributo* : Es la instanciación de una dominio en una relación. Se corresponde con el concepto de campo o columna de una tabla.
- *Tupla* : Cada uno de los conjuntos de valores de dominio para una instancia. Se corresponde con el concepto de registro o fila.
- *Valor de Dominio* : Valor presentado sobre un dominio, de un atributo para una determinada tupla.
- *Relación* : Elemento principal del modelo, se corresponde con el concepto de tabla.

En el modelo relacional, se define una **Relación** de la siguiente forma:

Dado un conjunto de n tipos o dominios $T_i (i = 1, 2, \ldots, n)$, no necesariamente distintos entre sí, r es una relación sobre dichos tipos si consta de dos partes: una cabecera (esquema, o intensión) y un cuerpo (extensión).

- La **cabecera** es un conjunto de n atributos de la forma $A_i : T_i$, donde los A_i, que son todos distintos, son los nombres de atributo de r y los T_i son los nombres de tipo correspondientes T_i para $(i = 1, 2, \ldots, n)$.

  Así pues la cabecera de una relación tendría la siguiente forma: $\{(A_1 : T_1), (A_2 : T_2), \ldots, (A_n : T_n)\}$

- El **cuerpo** de una relación es un conjunto de m tuplas t, en donde t es a su vez un conjunto de componentes de la forma $A_j : v_{jk}$ tal que $\{(A_1 : v_{11}), (A_2 : v_{21}), \ldots, (A_j : v_{jk})\}$ donde v_{jk} es un valor de tipo T_j para la fila k, es decir, el valor de atributo para el tipo $A_j (j = 1, 2, \ldots, n)$ de la tupla $t_k (k = 1, 2, \ldots, m)$.

A los valores n y m se les denomina **grado** y **cardinalidad** de la relación r, respectivamente. Mientras que el grado permanece constante, la cardinalidad varía con la inclusión o eliminación de tuplas.

A partir de esta definición se pueden realizar las siguientes reflexiones:

1. En términos de representación tabular de una relación, la cabecera es la fila de nombres de columna y de nombres de tipo correspondiente, la cual vendría a representar la definición intensional de la relación, mientras que el cuerpo es el conjunto de filas de datos, y por tanto la definición extensional.

2. En la misma relación o en relaciones distintas, diferentes A_i pueden estar definidos sobre el mismo T_i.

3. Pueden existir valores válidos para un tipo, y que estos no estén presentes en la base de datos.

3.2.2. Modelo Entidad-Relación

El modelo Entidad-Relación (ER) fue introducido en [Chen, 1976], aunque posteriormente otros autores [Teorey, 1989, Batini et al., 1992, Thalheim, 1993] han propuesto pequeñas variaciones y extensiones. El modelo ER es el modelo semántico más utilizado, y se ha convertido en un estándar, utilizado principalmente en la fase conceptual del diseño de una base de datos.

Para poder observar con mayor claridad la relación existente entre la Lógica Descriptiva y el modelo semántico Entidad-Relación, vamos a definir formalmente éste último.

Los elementos básicos del modelo ER son:

- *Entidades*: Denotan un conjunto de objetos, caracterizados por ser instancias de un determinado conjunto de propiedades.

- *Relaciones*: Denotan un conjunto de tuplas, también llamadas instancias, cada una de las cuales representa una asociación entre las diferentes entidades que participan en la relación.

- *Atributos*: Se emplean para modelar propiedades elementales. Los valores de los atributos pertenecen a dominios predefinidos, compuestos por tipos representables.

Aparte de los elementos básicos es necesario definir algunos conceptos adicionales.

- *Rol* : Dado que una entidad puede participar más de una vez en una misma relación, se introduce la noción de rol para representar e identificar la semántica de dicha participación. La aridad de una relación es el número de los roles que participan en ella.

- *Restricción de Cardinalidad* : Se asocia a un rol, para restringir el número de veces que una instancia de una entidad puede participar con ese rol en instancias de la relación. Las restricciones de cardinalidad se suelen utilizar en una forma reducida, donde la cardinaliad mínima es 0 o 1 y la máxima 1 o ∞.

- *Especialización* : Se dice que una entidad B es una especialización de otra entidad A, si todas las instancias de B son también instancias de A. Esto provoca la herencia de atributos por parte de una entidad a sus sub-entidades.

Con el objetivo de relacionar modelo ER y DLs es mejor emplear una descripción algo mas formal, y que abstraiga las características mas importantes presentes en las distintas variantes del modelo ER. Para ello, utilizaremos la definición formal y notación empleada en [Borgida et al., 2003].

Un esquema ER $\mathcal{S}$ se construye comenzando a partir de conjuntos disjuntos par a par de símbolos de entidades, símbolos de relaciones, símbolos de roles, símbolos de atributos y símbolos de dominio.

$$\mathcal{S}\langle E, R, U, A, D\rangle$$

Cada símbolo de dominio D tiene asociado un dominio básico predefinido $D^{\mathcal{B}_\mathcal{D}}$ y se asume que los dominios básicos son disjuntos par a par. Un símbolo de relación de aridad n tiene asociados n símbolos de rol, cada uno con su respectivo símbolo de entidad, y define una relación entre dichas entidades. Suponemos que cada rol pertenece a una única relación, de tal forma que determina una única entidad. Las restricciones de cardinalidad se representan mediante dos funciones aplicadas sobre los símbolos de rol: $cmin_\mathcal{S}$ que devuelve un entero no negativo y $cmax_\mathcal{S}$ que devuelve un valor incluido en el conjunto de los valores enteros positivos unión el símbolo especial ∞. Las relaciones de especialización entre entidades se modelan mediante una relación binaria $\preceq_\mathcal{S}$.

La semántica de un esquema ER puede darse indicando los estados de la base de datos que se consideran consistentes con la estructura de información dada por el esquema. Formalmente un estado de la base de datos $\mathcal{B}$ correspondiente a un esquema ER $\mathcal{S}$ está constituido por un conjunto finito no vacío $\Delta^\mathcal{B}$, que se asume es disjunto a todos los dominios básicos $\mathcal{B}_\mathcal{D}$ y una función $\cdot^\mathcal{B}$ que mapea:

- todo símbolo de dominio D con el correspondiente dominio básico $D^{\mathcal{B}_\mathcal{D}}$,
- toda entidad E con un subconjunto $E^\mathcal{B}$ de $\Delta^\mathcal{B}$,
- todo atributo A a un conjunto $A^\mathcal{B} \subseteq \Delta^\mathcal{B} \times \bigcup_{D \in \mathcal{D}_\mathcal{S}} D^{\mathcal{B}_\mathcal{D}}$, y
- toda relación R a un conjunto $R^\mathcal{B}$ de tuplas etiquetadas sobre $\Delta^\mathcal{B}$.

Una tupla etiquetada sobre un dominio $\Delta^\mathcal{B}$ es una función de un conjunto de roles en $\Delta^\mathcal{B}$. La tupla etiquetada T que mapea el rol U_i a o_i para $i \in \{1, \ldots, n\}$ se denota como $\langle U_1 : o_1, \ldots, U_n : o_n\rangle$. También se escribe $T[U_i]$ para denotar o_i y se denomina el componente-U_i de T. Los elementos $E^\mathcal{B}$, $A^\mathcal{B}$, $R^\mathcal{B}$ se denominan instancias de E, A y R respectivamente.

Un estado de la base de datos se considera aceptable si satisface todas las restricciones de integridad que forman parte del esquema. Formalmente se define la noción de estado legal de una base de datos de la siguiente forma:

Un estado de base de datos $\mathcal{B}$ es legal para un esquema ER $\mathcal{S}$, si satisface las siguientes condiciones:

- Para cada par de entidades E_1, E_2 con $E_1 \preceq_{\mathcal{S}} E_2$, se verifica que $E_1^{\mathcal{B}} \subseteq E_2^{\mathcal{B}}$.
- Para cada entidad E, si E tiene un atributo A con dominio D, entonces para cada instancia $e \in E^{\mathcal{B}}$ hay exactamente un elemento $a \in A^{\mathcal{B}}$ con e como primer componente, y el segundo componente de a es un elemento de $D^{\mathcal{B}_{\mathcal{D}}}$.
- Para cada relación R de aridad n entre entidades $E_1, \ldots, E_n$, para la cual R está conectada mediante los roles $U_1, \ldots, U_n$ respectivamente, todas las instancias de R son de la forma $\langle U_1 : e_1, \ldots, U_n : e_n \rangle$, donde $e_i \in E_i^{\mathcal{B}}, i \in \{1, \ldots, n\}$.
- Para cada rol U de la relación R asociada con la entidad E, y por cada instancia e de E, se verifica que $cmin_{\mathcal{S}}(U) \leq |\{r \in R^{\mathcal{B}} | r[U] = e\}| \leq cmax_{\mathcal{S}}(U)$. Esto es, la cardinalidad de un rol, o lo que es lo mismo, el número de tuplas que el rol tiene en la relación, debe ser mayor o igual que la cardinalidad mínima del rol y menor o igual que la cardinalidad máxima.

3.2.3. Base de Conocimiento DL

Una base de conocimiento DL, se define en [Baader and Werner, 2003] como un conjunto $\mathcal{T}$ de axiomas terminológicos, y un conjunto $\mathcal{A}$ de axiomas de aserción. Los axiomas en $\mathcal{T}$ describen hechos sobre conceptos C y roles R, mientras que los de $\mathcal{A}$ describen hechos acerca de las instancias individuales de los conceptos y roles. Esta definición se corresponde con lo que anteriormente hemos denominado *TBox* y *ABox*.

Una base de conocimiento terminológica $\mathcal{T}$ o *TBox* realiza aserciones acerca de cómo conceptos y roles se relacionan entre sí. Ésta consiste en un conjunto de axiomas de la forma $C \sqsubseteq D$, y $C \equiv D$ donde C y D son conceptos o clases. Una interpretación $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisface $\mathcal{T}$, si por cada axioma $(C \sqsubseteq D) \in \mathcal{T}, C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, y por cada axioma $(C \equiv D) \in \mathcal{T}, C^{\mathcal{I}} = D^{\mathcal{I}}$; $\mathcal{I}$ es satisfacible si existe alguna interpretación no vacía que la satisface.

Un rol o propiedad R de una *TBox* establece relaciones entre conceptos previamente definidos. Una restricción de valor para un rol se denota como $\forall R.C$, lo cual significa que todos los individuos de la relación R deben pertenecer a un concepto o clase C. Es posible indicar la cardinalidad de un rol o propiedad mediante las expresiones $\leq nR$ y $\geq nR$ donde n es la cardinalidad del rol.

La semántica asociada a las bases de conocimiento DL recibe una interpretación de teoría de conjuntos, un concepto o clase se interpreta como un conjunto de individuos y los roles o propiedades se interpretan como conjuntos de pares de individuos.

Veamos algunas propiedades de la *TBox*:

- **Subsunción** : Se dice que un concepto C es subsumido por otro concepto D con respecto a $\mathcal{T}$, escrito como $\mathcal{T} \models C \sqsubseteq D$, si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ en toda interpretación $\mathcal{I}$ que satisface $\mathcal{T}$

- **Satisfacibilidad** : Se dice que un concepto C es satisfacible con respecto a $\mathcal{T}$, escrito como $\mathcal{T} \models C \not\sqsubseteq \bot$ si $C^{\mathcal{I}} \neq \emptyset$ en alguna $\mathcal{I}$ que satisface $\mathcal{T}$.

- **Insatisfacibilidad** : Un concepto C se dice insatisfacible (no satisfacible) con respecto a $\mathcal{T}$, escrito como $\mathcal{T} \models \neg C$ si $C^{\mathcal{I}} = \emptyset$ en toda $\mathcal{I}$ que satisface $\mathcal{T}$.

- **Equivalencia** : Dos conceptos C y D son equivalentes con respecto a $\mathcal{T}$ si $C^{\mathcal{I}} = D^{\mathcal{I}}$ para cada interpretación $\mathcal{I}$ de $\mathcal{T}$. En este caso se denota $C \equiv_{\mathcal{T}} D$ o $\mathcal{T} \models C \equiv D$.

- **Disjunción** : Dos conceptos C y D son disjuntos con respecto a $\mathcal{T}$ si $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ para toda interpretación $\mathcal{I}$ de $\mathcal{T}$.

El segundo componente de una base de conocimiento es la descripción del mundo a través de las instancias que lo componen. Recordemos que lo habíamos designado como $\mathcal{A}$, y se corresponde con la $ABox$.

La $ABox$ describe un estado específico del dominio de la aplicación en términos de conceptos y roles. Estos conceptos y roles de la $ABox$ se corresponden con los definidos en la $TBox$. En la $ABox$ se introducen individuos, dándoles un nombre, y se pueden realizar aserciones sobre las propiedades de los individuos introducidos.

Los individuos en la $ABox$ se denotan con letras minúsculas $a, b, c, \ldots$. Utilizando los conceptos y roles definidos en la $TBox$, podemos realizar dos tipos de aserciones: $C(a)$, indicando que una instancia a pertenece a un concepto C (aserción sobre conceptos), y $R(b, c)$ indicando que la instancia b está relacionada con la instancia c a través del rol R (aserción sobre roles). Podemos decir, que una $Abox$ $\mathcal{A}$, no es más que un conjunto finito de aserciones de los tipos vistos anteriormente.

Empleando la interpretación anterior $\mathcal{I}$, podemos mapear cada nombre de individuo con un elemento $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. La interpretación $\mathcal{I}$ satisface las aserciones de conceptos $C(a)$ si $a^{\mathcal{I}} \in C^{\mathcal{I}}$ y, de forma análoga, la interpretación $\mathcal{I}$ satisface las aserciones de roles $R(a, b)$ si $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

Se dice que una interpretación $\mathcal{I}$ satisface la $ABox$ $\mathcal{A}$ si satisface cada aserción en $\mathcal{A}$. En ese caso se dice que $\mathcal{I}$ es un modelo de la aserción o de la $ABox$.

Para concluir, veremos en qué forma podemos relacionar ambos componentes de la base de conocimiento a través de la interpretación $\mathcal{I}$. Se dice que $\mathcal{I}$ satisface una aserción α o una $ABox$ $\mathcal{A}$ con respecto a una $TBox$ $\mathcal{T}$ si, además de ser un modelo de α o de $\mathcal{A}$, es un modelo de $\mathcal{T}$. Así pues, un modelo de $\mathcal{A}$ y $\mathcal{T}$ es una abstracción de un mundo concreto donde los conceptos son interpretados como subconjuntos del dominio como requiere la $TBox$ y donde la pertenencia de los individuos a conceptos y sus relaciones entre ellos en términos de roles verifica las aserciones en la $ABox$.

A partir de este punto identificaremos los roles de una base de conocimiento, como propiedades y los denotaremos como P, para evitar la confusión con los roles definidos en el contexto del modelo ER.

3.3. Propuesta de Almacenamiento de Ontologías OWL en SGBDOR

En esta sección profundizaremos en la propuesta de almacenamiento de ontologías OWL y sus instancias en Sistemas Gestores de Bases de Datos Objeto-Relacionales. La propuesta está motivada por la necesidad de gestionar de forma eficiente grandes volúmenes de datos de instancia, en un entorno orientado a la resolución de consultas, donde la capacidad de razonamiento sobre una ontología no es tan importante como la capacidad de dar respuesta a consultas en un tiempo apropiado.

Así pues, nuestra propuesta tiene dos vertientes claramente diferenciadas. Por un lado queremos realizar un acceso eficiente a los datos de instancia disponibles, y por otro, queremos conservar la estructura ontológica inicial para poder ofrecer ciertas capacidades básicas de razonamiento. Las distintas aproximaciones que vamos a adoptar a la hora de resolver cada uno de los diferentes aspectos del problema, van a condicionar la estructura de almacenamiento que vamos a emplear.

Parece obvio que para poder ofrecer un acceso lo más eficiente posible a las instancias, lo deseable sería diseñar un esquema de base de datos relacional apropiado para el problema a modelar. El problema en cuestión está codificado en la descripción intensional que conforma la ontología, es por esto que el primer objetivo es trasladar ese conocimiento intensional y expresarlo en forma de un esquema de base de datos. Para ello debemos realizar un paso intermedio consistente en expresar la mayor cantidad posible de dicho conocimiento intensional, en forma de un modelo semántico apropiado, que nos permita una posterior implantación en un sistema relacional. Así pues, transformaremos la información acerca del dominio del problema contenida en la ontología en un modelo semántico Entidad-Relación, lo cual nos va a permitir la posterior implantación de éste en una base de datos relacional. Como consecuencia directa de esta transformación el problema del diseño de un esquema de almacenamiento, se reduce a un proceso de diseño de una ontología de dominio OWL y su posterior transformación en un modelo semántico. El modelo semántico ER obtenido puede ser transformado directamente en un esquema de base de datos relacional.

Llegados a este punto disponemos de un esquema relacional de base de datos, en el que podemos almacenar los valores de instancia de la ontología. Sin embargo el proceso de transformación nos ha hecho perder semántica en cada transformación, esto dificulta en gran medida la recuperación de la información intensional original contenida en la ontología. Como consecuencia de esta pérdida de la información original, el acceso a los datos queda reducido al acceso normal a un esquema relacional, limitando la posibilidad de hacer consultas semánticas.

Para poder realizar consultas semánticas ricas, necesitaríamos conservar la información original de la ontología. Para evitar esta pérdida de la semántica del problema, proponemos almacenar también la ontología dentro de la base de datos. El almacenamiento de ésta se realizaría de forma que sus propiedades y relaciones definidas permaneciesen intactas, de modo que pudiésemos conocer la estructura original. Para guardar la parte intensional de la ontología, diseñaremos un esquema relacional que nos va a permitir almacenarla sin pérdida de información. Recordemos que en el Capítulo 2 ya realizamos una extensa

revisión de las diferentes aproximaciones empleadas para el almacenamiento de ontologías en bases de datos relacionales.

Así pues, por una parte tenemos la semántica intacta en la ontología almacenada, y por otro una representación adecuada para las instancias. Sin embargo necesitamos establecer algún tipo de conexión entre ambas partes, ya que las consultas semánticas realizadas sobre la ontología deben corresponderse con las instancias que las verifican. Para establecer una conexión entre ambas partes diseñamos un conjunto de tablas de metadatos, que van a permitir enlazar las descripciones de la ontología con las instancias que se corresponden con los objetos definidos.

La figura 3.1 muestra las estructuras de bases de datos empleadas para almacenar la ontología y sus instancias.

El esquema propuesto se divide en dos componentes:

- Esquema de Individuos: Este esquema es creado por el algoritmo de transformación para almacenar los individuos ($ABox$) de la ontología en un esquema de base de datos creado a partir de ésta.
- Tablas de Ontología del Catálogo del Sistema: Almacenan la ontología ($TBox$) y las relaciones entre los individuos de la ontología.

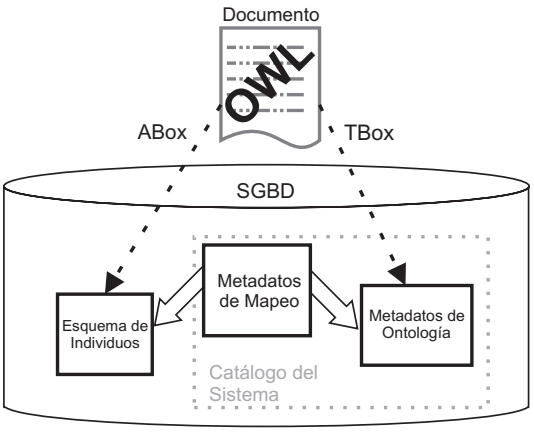


Figura 3.1: Esquema de Ontología e Individuos

El Esquema de Individuos es el esquema resultante de la transformación de un documento OWL en el esquema relacional de base de datos. Cada documento OWL creará un esquema diferente dependiendo de la ontología de dominio que representa. En el proceso de generación del esquema, se realiza una transformación de OWL a un modelo semántico, en este caso ER, y posteriormente este se transforma en un modelo relacional aplicando las reglas de transformación conocidas [Date, 1990]. De esta forma, el proceso completo permite que las clases se transformen en tablas, las propiedades de objetos en relaciones, las propiedades de tipo de dato en atributos de tablas, y así sucesivamente.

Los nombres de tablas y de atributos se generan de forma automática usando el nombre de la clase, y utilizando como prefijo el nombre de la ontología a la cual pertenece la clase.

Las Tablas de Ontología del Catálogo del Sistema contienen dos estructuras diferentes:

- *Metadatos de Ontologías*: Este conjunto de tablas almacena los diferentes elementos de la ontología, términos, propiedades, relaciones, restricciones...
- *Metadatos de Mapeo*: Estas tablas almacenan los datos necesarios para establecer un enlace entre un individuo en el Esquema de Individuos y el concepto de la ontología al cual pertenece dicho individuo.

A continuación detallaremos cada uno de los componentes del esquema propuesto y definiremos las transformaciones necesarias para expresar la ontología OWL como un modelo ER y posteriormente transformar éste en un esquema relacional.

3.3.1. De OWL a Esquema Relacional de Base de Datos

Antes de presentar nuestra propuesta recordaremos brevemente algunos de los trabajos que tratan con el problema del almacenamiento de ontologías OWL en esquemas de bases de datos.

En [Vysniauskas and Nemuraite, 2006] se presentan los detalles básicos de la transformación de OWL a esquema relacional y se presenta un breve ejemplo. La transformación primero convierte las clases, luego las propiedades de objeto, las propiedades de tipo de dato y las restricciones. Una vez definido el esquema se almacenan las instancias de la ontología en las tablas adecuadas. De forma parecida en [Astrova et al., 2007] se proponen un conjunto de reglas para convertir ontologías OWL a esquemas de base de datos. Estas y otras propuestas han sido analizadas con detalle en la Sección 2.1.2.

Estos trabajos proponen una transformación directa desde la ontología al modelo relacional, sin embargo existe un proceso intermedio que se obvia, que es la representación mediante un modelo semántico.

Para poder lograr que la migración de datos desde ontologías OWL a bases de datos relacionales se realice sin incidentes, algunos autores [Bagui, 2009] proponen la transformación de ontologías a modelos semánticos ER. El uso de un paso intermedio en el que se genera un modelo ER es útil desde el punto de vista de la documentación del esquema de base de datos final generado, así como es útil para la formalización de las diferentes transformaciones.

El paso final consiste en la transformación de un esquema ER a un esquema relacional. Este proceso es bien conocido y sencillo, puede encontrarse definido en los siguientes trabajos [Chen, 1976, Bagui and Earp, 2003, Navathe and Elmasri, 2007].

Transformación OWL a ER

En esta sección trataremos la transformación de ontologías OWL al modelo semántico ER, desde un punto de vista formal. Veremos las transformaciones necesarias para expresar la ontología como un modelo ER, que posteriormente será expresado como un esquema relacional, el cual formará parte del esquema de almacenamiento que proponemos.

Para la transformación de OWL a ER tendremos en cuenta las líneas maestras para el mapeo de OWL a ER propuestas por OMG a través del Ontology Definition Metamodel en [OMD, 2005], a pesar de que esta propuesta no llegó a formar parte de la versión final del documento. La tabla 3.1 presenta el conjunto de transformaciones recomendadas, tal y como aparecen en la versión más reciente del borrador que incluía mapeos de OWL a ER.

Tabla 3.1: Mapeo de OWL a ER

OWL Model	**ER Model**
RDFSResource	Elemento con Nombre
OWLOntology	Modelo
OWLClass	Entidad
OWLRestriction	Ver OWLDatatypeProperty y OWLObjectProperty
OWLDatatypeProperty	Atributo
OWLObjectProperty	Relación
OWLDataRange	Dominio Atómico

Teniendo en cuenta que al comienzo de este capítulo establecimos que una ontología trataba con la definición intensional de un problema y, dado que el esquema ER es adecuado para la representación de este tipo de información, pero no tiene la capacidad de representar información extensional de ningún tipo, estableceremos la relación entre una ontología y un esquema a través de una aplicación.

Definimos la siguiente aplicación de una ontología en un esquema ER.

$$f : \mathcal{O} \longrightarrow \mathcal{S}$$

o bien

$$f : \mathcal{O}\langle\mathcal{T}\rangle \longrightarrow \mathcal{S}\langle E, R, U, A, D\rangle$$

Clases Las clases definidas en la ontología, se representarán como entidades en el modelo ER. Un concepto C_i se traslada a un esquema ER como una entidad E_i.

$$C_i \longmapsto E_i$$

Un caso particular son las subclases, las subclases de la ontología se mostrarán en el modelo ER como entidades, que a su vez son especializaciones. Un concepto C_i que a su vez es subconcepto de un concepto C_j se mapea en ER como una entidad E_i que es una especialización de la entidad mas general E_j.

$$C_i \sqsubseteq C_j \longmapsto E_i \preceq_{\mathcal{S}} E_j$$

Las clases pueden estar definidas mediantes expresiones complejas, este tipo de clases complejas se verán en el apartado de restricciones.

Propiedades de Tipo de Dato Las propiedades de tipo de dato se transforman en atributos de la entidad obtenida a partir de la clase que tienen como dominio, y se mapean al correspondiente tipo básico.

Dado un rol R_i cuyo dominio es el concepto C_i definido en $\mathcal{O}\langle\mathcal{T}\rangle$ este se mapea en el esquema ER $\mathcal{S}$ como un atributo $A_j \in A$ asociado a una entidad $E_i \in E$ con un dominio $D_k \in D$ cuyo dominio básico predefinido es $D^{\mathcal{B}_\mathcal{D}}$.

Propiedades de Objeto Las propiedades de objetos de una ontología OWL relacionan dos clases a través de una propiedad o rol. Así pues éstas se modelan como relaciones entre clases del rango y el dominio (entidades en el ER) de la propiedad. Atendiendo a la cardinalidad de estas propiedades en la ontología, podemos determinar la participación y cardinalidad de los roles en la relación representada en el modelo ER.

Dada una propiedad P_i perteneciente a una ontología $\mathcal{O}$ cuyo rango es un concepto C_r y su dominio un concepto C_d, ésta se representará en un esquema entidad-relación $\mathcal{S}$ como una relación R_i entre las entidades E_r y E_d (obtenidas al mapear los conceptos C_r y C_d respectivamente), unidas por un rol U_j.

Para una propiedad de objeto dada, si la cardinalidad mínima del rango de la propiedad es 0, podemos indicar que la participación es parcial, y para un valor de 1 o superior, la participación se considera total.

La cardinalidad mínima de una propiedad P_i denotada como $\geq nP_i$ se mapea como la cardinalidad mínima del rol U_j, denotada como $cmin_\mathcal{S}(U_j) = n$.

De modo similar si la cardinalidad máxima del rango de la propiedad es 1, podemos decir que su cardinalidad es de 1, mientras que si es mayor que 1 decimos que la cardinalidad para esa relación en el ER es de n.

La cardinalidad máxima de una propiedad P_i denotada como $\leq nP_i$ se mapea como la cardinalidad máxima del rol U_j, denotada como $cmax_\mathcal{S}(U_j) = n$.

Las distintas combinaciones de participación y cardinalidad darán lugar a relaciones $1:1$, $1:n$, $n:n$. Atendiendo a los roles implicados en la relación y a sus cardinalidades se realizará el correspondiente paso al modelo relacional.

Algoritmo completo de transformación OWL a Esquema Relacional

Tras la definición inicial de una ontología de dominio en OWL, es necesario definir un esquema de base de datos apropiado para albergar sus instancias. La creación de este esquema se realizará a través de un algoritmo de transformación, que tomará la ontología generando el modelo semántico correspondiente y creará la estructura correspondiente en una base de datos relacional.

El esquema de Individuos presentado anteriormente, en el que se almacenan los datos de instancia, es el resultado de aplicar el algoritmo de transformación a la ontología OWL. Este proceso consta de la transformación de la ontología en un modelo ER y posteriormente el paso de éste a tablas. Las tablas y definiciones de relaciones se crean en memoria y tras la finalización del proceso el esquema resultante se traslada a sentencias *SQL*.

Presentaremos el algoritmo en pseudocódigo con la finalidad de mostrar detalladamente el proceso seguido en la transformación. Aunque la sintaxis utilizada es bastante sencilla,

explicaremos algunos detalles que pueden ser de utilidad.

El elemento `list` es una lista en el que se irán incluyendo los distintos elementos, estas listas se definirán como colas FIFO, en las que el primer elemento introducido es el primero en sacarse de la lista. Utilizaremos el comando `NEXT` para obtener el siguiente elemento de una lista. `DBSchema` es un elemento que contiene un conjunto de elementos tabla, representando el esquema de base de datos relacional a crear. El operador de asignación que empleamos es `<-`, dependiendo de si se usa antes de un comando `ADD` o uno `CREATE`, añadirá o creará un nuevo elemento al elemento al que ha sido asignado. Haremos referencia a los atributos de un elemento empleando la sintaxis `elemento.atributo`. El comando `GET elemento id FROM elemento_complejo` busca un objeto de tipo `elemento` que tenga el `id` especificado, dentro de un elemento complejo. Puede usarse para recuperar tablas usando su identificador, que se encuentran almacenadas en el elemento `DBSchema`.

A continuación procedemos a la descripción del algoritmo.

El primer paso en la ejecución del algoritmo es la creación de las tablas. Se recorre la ontología a partir de su clase raíz, cada vez que se procesa una clase, si ésta tiene subclases, éstas se añaden a la lista de procesamiento. Las clases de la ontología se transforman en tablas, estableciendo relaciones de acuerdo a propiedades de herencia tales como `rdfs:subClassOf`. Las subclases generan nuevas tablas con referencias a la tabla derivada a partir de su superclase. Las columnas que conforman la Clave Primaria (PK) se etiquetan con el nombre de la clase a partir de la cual se han generado. El proceso en pseudocódigo puede verse en el Algoritmo 1.

El siguiente paso consiste en la transformación de `owl:ObjectProperty`, las propiedades de objeto, para representar relaciones entre las tablas que ya han sido definidas. Dependiendo de factores tales como la cardinalidad, se añadirán nuevas tablas o atributos al esquema. Como regla general, las propiedades de objeto multi-valuadas describiendo relaciones muchos a muchos, se mapean como tablas nuevas con Claves Externas (FK) a las tablas definidas en `rdfs:domain` y `rdfs:range`, actuando ambas como clave primaria de la relación. Las propiedades de objeto mono-valuadas se mapean en la tabla indicada por `rdfs:domain` como claves externas a la tabla indicada en `rdfs:range`. Existen algunas excepciones a esta regla, como se puede ver en la especificación del Algoritmo 2.

Las propiedades de tipo de datos se transforman en atributos de tablas. El tipo de las columnas creadas por mapeos de `owl:DatatypeProperty` concuerda con el tipo de datos descrito en `rdfs:range`. Este tipo de datos es un tipo de dato básico XML Schema válido, todos estos tipos pueden mapearse a tipos *SQL* de forma sencilla.

La serialización en XML de OWL emplea los tipos definidos para XML Schema (XSD), a continuación veremos los mapeos de los tipos XSD a tipos SQL.

La Tabla 3.2 muestra el mapeo de los tipos de cadena de XSD en SQL.

La Tabla 3.3 presenta el mapeo de los tipos binarios presentes en XML Schema a SQL.

La Tabla 3.4 muestra como los valores numéricos se mapean a SQL. La tabla muestra el tipo básico XSD, el tipo por defecto que se asignaría en SQL, y otros tipos SQL que son compatibles con el tipo básico XSD.

La Tabla 3.5 muestra los tipos de hora/fecha en XSD, el tipo por defecto al que se asigna en SQL, y otros posibles tipos SQL compatibles.

Finalmente, la Tabla 3.6 muestra otros tipos empleados en XSD que aunque no son

Algoritmo 1 Generación de Tablas

```
list IS FIFO LIST;
list <- ADD Root Classes;

DBschema <- EMPTY;

WHILE list IS NOT EMPTY

    class <- NEXT list;
    table <- CREATE TABLE;
    table.name <- class.rdf:ID;

    IF class IS ROOT THEN
        column <- CREATE PK COLUMN;
        column.name <- class.rdf:ID;
        table <- ADD column;
    ELSE
        column <- CREATE PK COLUMN;
        column.name <- class.rdf:ID;
        parentTable <- GET TABLE class.parentClass.rdf:ID FROM DBschema;
        column <- ADD FK REFERENCE TO parentTable.PK;
        table <- ADD column;
    END IF;

    IF class HAS SUBCLASS THEN
        IF class.subClass IS owl:disjointWith ONE OR MORE SIBLINGS THEN
            list <- ADD subClass;
        END IF;
    END IF;

    DBschema <- ADD table;

END WHILE;
```

Algoritmo 2 Generación de Relaciones

```
list IS FIFO LIST;
list <- ADD ALL Root owl:ObjectProperty;
DBschema HAS CONTENT;

WHILE list IS NOT EMPTY
    property <- NEXT list;
    domain <- property.rdfs:domain;

    IF domain.owl:cardinality == 1 OR domain.owl:maxCardinality == 1 THEN
        IF property IS SUBPROPERTY THEN
            column <- CREATE COLUMN;
            column.name <- property.rdf:ID;

            IF property.rdf:type.rdf:resource IS owl:TransitiveProperty THEN
                column <- ADD UNIQUE CONSTRAINT;
            END IF;

            IF property.owl:cardinality == 1 OR
               property.owl:minCardinality == 1 THEN
                   column <- ADD NOT NULL CONSTRAINT;
            END IF;

            tableParentRange <- GET TABLE property.parentProperty.rdfs:range.rdf:resource
                                FROM DBschema;

            column <- ADD FK REFERENCE TO tableParentRange.FK
            tableRange <- GET TABLE property.rdfs:range.rdf:resource FROM DBschema;
            tableRange <- ADD column;
            DBschema <- UPDATE tableRange;
        ELSE
            column <- CREATE COLUMN;
            column.name <- property.rdf:ID;

            IF property.rdf:type.rdf:resource IS owl:TransitiveProperty THEN
                column <- ADD UNIQUE CONSTRAINT;
            END IF;

            IF property.owl:cardinality == 1 OR property.owl:minCardinality THEN
                column <- ADD NOT NULL CONSTRAINT;
            END IF;

            IF property.rdf:type.rdf:resource IS owl:TransitiveProperty THEN
                column <- ADD UNIQUE CONSTRAINT;
            ENd IF;

            tableRange <- GET TABLE property.rdfs:range.rdf:resource FROM DBschema;
            column <- ADD FK REFERENCE TO tableRange.FK
            tableDomain <- GET TABLE property.rdfs:domain.rdf:resource FROM DBschema;
            tableDomain <- ADD column;
            DBschema <- UPDATE tableDomain;
        END IF;
    ELSE
        IF property IS SUBPROPERTY THEN

            table <- CREATE TABLE;
            table.name <- property.rdf:ID;
            columnParentRange <- CREATE PK COLUMN;
            tableParentRange <- GET TABLE property.parentProperty.rdfs:range.rdf:resource
                                FROM DBschema;

            columnParentRange.name <- tableParentRange.FK.name;
            columnParentRange <- ADD FK REFERENCE TO tableParentRange.FK;
            columnRange <- CREATE PK COLUMN;
            tableRange <- GET TABLE property.rdfs:range.rdf:resource FROM DBschema;

            columnRange.name <- tableRange.FK.name;
            columnRange <- ADD FK REFERENCE TO tableRange.FK;
            table <- ADD columnParentRange;
            table <- ADD columnRange;
            DBschema <- ADD table;
        ELSE
            table <- CREATE TABLE;
            table.name <- property.rdf:ID;
            columnDomain <- CREATE PK COLUMN;
            tableDomain <- GET TABLE property.rdfs:domain.rdf:resource FROM DBschema;

            columnDomain.name <- tableDomain.FK.name;
            columnDomain <- ADD FK REFERENCE TO tableDomain.PK;
            columnRange <- CREATE PK COLUMN;
            tableRange <- GET TABLE property.rdfs:range.rdf:resource FROM DBschema;

            columnRange.name <- tableRange.FK.name;
            columnRange <- ADD FK REFERENCE TO tableRange.PK;
            table <- ADD columnDomain;
            table <- ADD columnRange;
            DBschema <- ADD table;
        END IF;
    END IF;

    IF property HAS SUBPROPERTIES THEN
        list <- ADD subProperties;
    END IF;
END WHILE;
```

Tabla 3.2: Mapeo de Tipos Cadena de XML Schema a SQL

Tipo XSD	Tamaño	Tipo de Dato SQL por Defecto	SQL Compatible
string	n	VARCHAR2(n) si n <4000, si no VARCHAR2(4000)	CHAR, CLOB
string	-	VARCHAR2(4000) si mapUnboundedStringToLob = "false", CLOB	CHAR, CLOB

Tabla 3.3: Mapeo de Tipos Binarios de XML Schema (hexBinary/base64Binary) a SQL

Tipo binario XSD	Tamaño	Tipo SQL por defecto	SQL Compatible
hexBinary, base64Binary	n	RAW(n) si n <2000, si no RAW(2000)	RAW, BLOB
hexBinary, base64Binary	-	RAW(2000) si mapUnboundedStringToLob = "false", BLOB	RAW, BLOB

de uso común, pueden aparecer como dominio de las propiedades de tipo de dato en las ontologías. En la tabla se muestra el tipo básico XSD, el tipo SQL por defecto al que se mapea, y otros tipos SQL compatibles.

A continuación, el Algoritmo 3 muestra el pseudocódigo para la transformación de propiedades de tipo de dato OWL en atributos de tabla de un esquema relacional, las tablas mostradas anteriormente se utilizan en la función `SQLTYPE()` para determinar el tipo SQL que corresponde al tipo XSD definido en la serialización XML de la ontología OWL.

Algoritmo 3 Generación de Atributos

```
list IS FIFO LIST;
list <- ADD ALL owl:DatatypeProperty;

DBschema HAS CONTENT;

WHILE list NOT EMPTY

    property <- NEXT list;
    domainTable <- GET TABLE property.rdfs:domain.name FROM DBschema;
    column <- CREATE COLUMN WITH TYPE SQLTYPE(property.rdfs:range.rdf:resource);
    column.name <- property.rdf:ID;
    domainTable <- ADD column;

END WHILE;
```

Las restricciones sobre propiedades son un tipo especial de descripción de clase. Describen una clase anónima, concretamente una clase que contiene todos los individuos que verifican la restricción. OWL distingue entre dos tipos de restricciones: restricciones de valores y restricciones de cardinalidad. Las restricciones de cardinalidad ya se han tratado, concretamente en el mapeo de propiedades `owl:ObjectProperty`. Las restricciones sobre valores se definen a través de `owl:someValuesFrom`, `owl:allValuesFrom` y `owl:hasValue`.

La restricción `owl:allValuesFrom` requiere que, para cada instancia de la clase que

Tabla 3.4: Mapeo de Valores Numéricos de XML Schema a SQL

Tipo XSD	**SQL por defecto**	**Tipos Compatibles SQL**
float	NUMBER	FLOAT, DOUBLE, BINARY_FLOAT
double	NUMBER	FLOAT, DOUBLE, BINARY_DOUBLE
decimal	NUMBER	FLOAT, DOUBLE
integer	NUMBER	NUMBER
nonNegativeInteger	NUMBER	NUMBER
positiveInteger	NUMBER	NUMBER
nonPositiveInteger	NUMBER	NUMBER
negativeInteger	NUMBER	NUMBER
long	NUMBER(20)	NUMBER
unsignedLong	NUMBER(20)	NUMBER
int	NUMBER(10)	NUMBER
unsignedInt	NUMBER(10)	NUMBER
short	NUMBER(5)	NUMBER
unsignedShort	NUMBER(5)	NUMBER
byte	NUMBER(3)	NUMBER
unsignedByte	NUMBER(3)	NUMBER

tiene instancias de la propiedad especificada, los valores de la propiedad sean todos miembros de la clase indicada por la cláusula `owl:allValuesFrom`. Este mapeo se realiza añadiendo una restricción de clave externa en la columna generada anteriormente por la propiedad, apuntando a la columna de la clave primaria de la clase especificada por `rdf:resource`.

En el caso de `owl:someValuesFrom`, al menos una de las propiedades de las instancias de la clase debe apuntar a un individuo que es parte de la clase indicada en la restricción. Como puede apreciarse, este caso es mucho mas complejo porque si la propiedad es multi-valuada la consistencia debe comprobarse mediante el uso de un disparador. Si es mono-valuada el valor para la propiedad debe ser una de las clases en la restricción, haciendo este caso particular equivalente al de `owl:allValuesFrom` presentado anteriormente.

También la restricción `owl:hasValue` es un caso complejo. Esta propiedad relaciona una clase de restricción a un valor, que puede ser un individuo o un dato. En el caso de que el valor sea un dato, la restricción se puede modelar en la definición *SQL* de la tabla empleando una restricción `CHECK`. Los problemas aparecen cuando el valor es un individuo, entonces es necesario utilizar un disparador. Toda la información que necesita el disparador para operar está disponible en la tabla de Metadatos de Mapeo.

Por ahora vamos a centrarnos en los casos simples en los que no es necesaria la definición de un disparador, el caso más complejo se tendrá en cuenta en futuros trabajos. El proceso que realiza el mapeo de restricciones se muestra en el Algoritmo 4.

Finalmente solo nos queda introducir las instancias en sus correspondientes tablas. Es

Algoritmo 4 Generación de Restricciones

```
list IS FIFO LIST;
list <- ADD Root Properties;

DBschema IS HAS CONTENT;

WHILE list NOT EMPTY

    property <- NEXT list;

    IF property HAS owl:Restriction THEN

       CASE property.restriction OF

        owl:allValuesFrom :
            tableDomain <- GET TABLE property.owl:domain FROM DBschema;
            tableRange <- GET TABLE property.owl:range FROM DBschema;
            column <- GET COLUMN property.rdf:ID FROM tableDomain;
            column <- ADD FK REFERENCE TO tableRange.PK;
            tableDomain <- UPDATE column;
            DBschema <- UPDATE tableDomain;

        owl:someValuesFrom:
            IF property.owl:cardinality == 1 OR
                        property.owl:maxCardinality ==1 THEN
                tableDomain <- GET TABLE property.owl:domain FROM DBschema;
                tableRange <- GET TABLE property.owl:range FROM DBschema;
                column <- GET COLUMN property.rdf:ID FROM tableDomain;
                column <- ADD FK REFERENCE TO tableRange.PK;
                tableDomain <- UPDATE column;
                DBschema <- UPDATE tableDomain;
            END IF;

        owl:hasValue:
            IF property.owl:hasvalue.rdfs:resource IS A VALUE
                table <- GET TABLE property.owl:domain FROM DBschema;
                column <- GET COLUMN property.rdf:ID FROM table;
                column <- ADD CHECK CONSTRAINT property.owl:hasvalue.rdfs:resource;
                table <- UPDATE column;
                DBschema <- UPDATE table;
            END IF;
    END IF;

    IF property HAS SUBPROPERTIES THEN
        list <- ADD property.subProperties;
    END IF;

END WHILE;
```

Tabla 3.5: Mapeo de Tipos Fecha/Hora de XML Schema a SQL

Fecha/hora XSD	**SQL por defecto**	**Tipos SQL Compatibles**
dateTime	TIMESTAMP	TIMESTAMP WITH TIME ZONE, DATE
time	TIMESTAMP	TIMESTAMP WITH TIME ZONE, DATE
date	DATE	TIMESTAMP WITH TIME ZONE
gDay	DATE	TIMESTAMP WITH TIME ZONE
gMonth	DATE	TIMESTAMP WITH TIME ZONE
gYear	DATE	TIMESTAMP WITH TIME ZONE
gYearMonth	DATE	TIMESTAMP WITH TIME ZONE
gMonthDay	DATE	TIMESTAMP WITH TIME ZONE
duration	VARCHAR2(4000)	ninguno

importante tener en cuenta que una instancia puede estar separada en diferentes tablas como resultado del mapeo realizado. Por cada instancia, los valores se almacenan en sus columnas correspondientes. Se accede a la base de datos como a cualquier otra base de datos relacional, toda la información del esquema puede ser consultada y modificada.

Cuando la definición de la ontología también se encuentra almacenada en la base de datos, los individuos y la ontología se encuentran relacionados a través de los Metadatos de Mapeo. Esta relación permite la creación de procedimientos y operadores para realizar tareas de razonamiento limitadas, tales como subsunción de clases. El uso de estos operadores y procedimientos debe hacerse de forma explícita en las consultas.

3.3.2. Esquema de Base de Datos para el Almacenamiento de la Ontología

En esta sección detallaremos el proceso empleado para obtener el esquema de almacenamiento de ontologías en base de datos. Tras analizar las diferentes propuestas para el almacenamiento de ontologías en esquemas relacionales vistas en la Sección 2.1.4, finalmente nos hemos decantado por emplear un esquema de tabla vertical en el que incluiremos los términos de la ontología, y una serie de tablas auxiliares externas que proporcionan información adicional acerca del término almacenado en la tabla principal. Un resumen de la estructura presentada puede encontrarse en [Barranco et al., 2007].

Los principales elementos de OWL se reflejan en el esquema, exceptuando aquellos relacionados con el control de versiones de ontologías, que se tratarán en futuros trabajos.

Los Metadatos de Ontologías se organizan tal y como se muestra en la figura 3.2. A continuación mostramos una descripción detallada de cada una de las tablas:

La tabla `Ont_NameSpaces` incluye información acerca del espacio de nombres definido para cada uno de los términos almacenados y el prefijo utilizado. Un mismo espacio de nombres puede utilizarse con distintos prefijos. Cada termino de la ontología tiene asociado su correspondiente espacio de nombres para poder distinguir entre elementos definidos en

Tabla 3.6: Mapeo de Otros Tipos Primitivos y Derivados de XML Schema a SQL

Tipo XSD	**Tipo SQL por defecto**	**Tipos SQL Compatibles**
boolean	RAW(1)	VARCHAR2
language(string)	VARCHAR2(4000)	CLOB, CHAR
NMTOKEN(string)	VARCHAR2(4000)	CLOB, CHAR
NMTOKENS(string)	VARCHAR2(4000)	CLOB, CHAR
Name(string)	VARCHAR2(4000)	CLOB, CHAR
NCName(string)	VARCHAR2(4000)	CLOB, CHAR
ID	VARCHAR2(4000)	CLOB, CHAR
IDREF	VARCHAR2(4000)	CLOB, CHAR
IDREFS	VARCHAR2(4000)	CLOB, CHAR
ENTITY	VARCHAR2(4000)	CLOB, CHAR
ENTITIES	VARCHAR2(4000)	CLOB, CHAR
NOTATION	VARCHAR2(4000)	CLOB, CHAR
anyURI	VARCHAR2(4000)	CLOB, CHAR
anyType	VARCHAR2(4000)	CLOB, CHAR
anySimpleType	VARCHAR2(4000)	CLOB, CHAR
QName	XDB.XDB$QNAME	ninguno
normalizedString	VARCHAR2(4000)	ninguno
token	VARCHAR2(4000)	ninguno

diferentes ontologías. Las URIs que identifican cada uno de los elementos/términos en una ontología se componen concatenando la URL de su *namespace* y el nombre del término en cuestión.

`Ont_Ontologies` almacena información sobre las diferentes ontologías. Cada ontología debe describirse adicionalmente como un término para que pueda ser asociada con el correspondiente espacio de nombres.

La tabla `Ont_Terms` contiene la representación de varios conceptos descritos en la ontología. Ontologías, clases y propiedades se almacenan aquí como elementos básicos que pueden ser utilizados en construcciones mas complejas. Cada término tiene asociado una ontología y un espacio de nombres, puesto que un elemento puede aparecer en una determinada ontología, pero estar definido en otro espacio de nombres. Además de estos elementos básicos, en esta tabla se definen dos elementos complejos `ClassList` que representa una lista de clases y `Group` que representa un conjunto de individuos. Las estructuras básicas de OWL deben definirse inicialmente antes de incluir ninguna ontología, esto es preciso para establecer relaciones entre términos como se verá mas adelante.

`Ont_ObjectProperties` contiene información sobre propiedades de objetos y sus características. El dominio y rango de la propiedad se almacenan como términos en la tabla

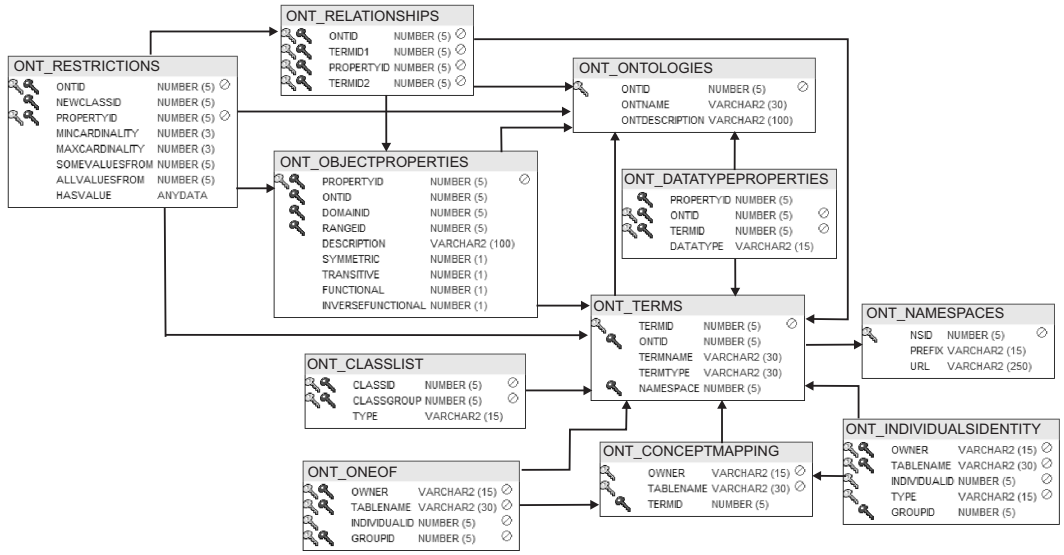


Figura 3.2: Esquema de Metadatos del Catálogo del Sistema

`Ont_Terms`. El resto de atributos incluyen la descripción textual de la propiedad, y todas sus posibles características. Las propiedades de objetos deben incluirse como términos en la tabla `Ont_Terms` para poder ser usados como parte de las relaciones.

La tabla `Ont_Restrictions` incluye información sobre restricciones de propiedades y las clases anónimas que se crean cuando se aplican las restricciones sobre la propiedad. Los atributos `mincardinality` y `maxcardinality` contienen valores numéricos de cardinalidad positivos. Los atributos `somevaluesfrom` y `allvaluesfrom` contienen un identificador de término correspondiente a una clase definida en la tabla `Ont_Terms`.

`Ont_Relationships` contiene información sobre las relaciones existentes entre dos términos de una ontología. Las relaciones básicas de OWL tales como `SubClassOf` pueden expresarse aquí. Los atributos `termID` apuntan a términos definidos en la tabla `Ont_Terms`. Estos términos pueden ser términos de clase, términos de propiedad o términos de lista de clases.

`Ont_DataTypeProperties` contiene información sobre propiedades de tipo de dato tales como el identificador de la propiedad, la ontología a la que pertenece, y el dominio y rango del tipo de dato. El dominio es un término en `Ont_Terms`. El atributo *Datatype* especifica el nombre del tipo que puede ser cualquiera de los definidos en el `xsd` XML Schema.

Las tablas de Metadatos de Mapeo relacionan los individuos en el Esquema de Individuos con las tablas de Metadatos de la Ontología donde se definen los conceptos apropiados. Estas tablas se definen de la siguiente forma:

`Ont_ConceptMapping` relaciona una tabla, y por tanto sus tuplas (individuos) en el Esquema de Individuos con su correspondiente clase en `Ont_Terms`. Una clase se representa

como una tabla para contener a sus individuos y como un término para permitir tareas de concordancia semántica o de razonamiento.

La tabla `Ont_IndividualsIdentity` contiene información relacionada con aspectos concernientes a la identidad de los individuos a través de diferentes ontologías. OWL define un conjunto de propiedades para gestionar la identidad de los individuos `owl:sameAs`, `owl:differentFrom` y `owl:AllDifferent`. Estas propiedades relacionan individuos, no clases. Es por esto que los tratamos en una tabla de sistema aparte, en la que se enlazan instancias y definiciones en una ontología. El atributo `type` describe el tipo de propiedad OWL empleada, una de las tres mencionadas anteriormente, y `groupID` es el identificador de grupo que contiene los individuos que están relacionados mediante la propiedad. La definición de un grupo de individuos también debe incluirse como término en la tabla `Ont_Terms`.

OWL proporciona la propiedad `owl:oneOf` para especificar una clase a través de una enumeración directa de sus miembros. `Ont_OneOf` contiene información acerca de los individuos pertenecientes a una enumeración particular. La propiedad `owl:oneOf` relaciona una clase con los individuos que la describen. Para poder reflejarlos en el catálogo del sistema es necesario incluir una relación entre la clase y un grupo de individuos. `owl:oneOf` debería estar presente como propiedad OWL en el Esquema de Metadatos, y la clase y grupo, deberían haberse insertado previamente en la tabla `Ont_Terms`. Una vez se cumplen estas precondiciones la lista de individuos pertenecientes al grupo se especifica en `Ont_OneOf`.

OWL también proporciona operadores tales como `owl:unionOf` y `owl:intersectionOf` para definir clases complejas basándose en operaciones realizadas sobre clases previamente definidas. Las clases construidas empleando estos operadores son como definiciones, en las que la extensión de la clase consiste exactamente en los individuos obtenidos en la operación. `Ont_ClassList` contiene la definición de las listas de clases. `owl:unionOf` y `owl:intersectionOf` deberían estar presentes como propiedades OWL definidas en el Esquema de Metadatos de Ontologías, para poder crear una relación entre una clase y la definición de una lista de clases utilizando estas propiedades. Una vez la relación está definida, la nueva clase se crea como una vista sobre las clases en el grupo en el que se encuentran almacenadas como tablas.

3.3.3. Gestión de Datos Difusos

Existen multitud de trabajos que tratan acerca del manejo de datos difusos en ontologías. Principalmente estos trabajos se centran en la descripción de conceptos y relaciones difusas, y en relaciones jerárquicas difusas. Este tipo de tratamiento supone un incremento de la complejidad en cuanto a la representación y tratamiento de este tipo de datos. En nuestra opinión también es muy importante, y sin duda más sencillo tratar la incertidumbre al nivel de atributos o de tipos de datos, por ejemplo, podemos decir que la edad de una persona está entre 25 y 30 años, o que es aproximadamente 30 años. Este tipo de representación de los datos está más ligada a los datos de instancia que a la propia representación de la ontología. Dado que nuestra propuesta distingue de forma clara el

tratamiento de la estructura de la ontología del almacenamiento y gestión de sus instancias, podemos añadir semántica a los datos utilizando las capacidades que proporciona el SGBDOR difuso sobre el que trabajamos [Cubero et al., 2004, Barranco et al., 2008b]. Este modelo de base de datos soporta la definición de tipos de datos difusos (la información detallada sobre este modelo se encuentra en la sección A.3.2 del Apéndice A), con sólo definir atributos como del tipo *OAFT object*, el cual puede contener valores numéricos difusos. Todo el tratamiento de los datos difusos corre por cuenta del SGBDOR difuso subyacente.

El uso de lógica difusa en las consultas, permite a los usuarios realizar búsquedas significativas expresando criterios flexibles. En [Barranco et al., 2005b] se usa la consulta sobre datos difusos para mejorar los resultados de las consultas, particularmente cuando se emplean condiciones múltiples. Las consultas flexibles permiten a los usuarios expresar sus necesidades de información de una forma intuitiva y flexible. Los resultados de las consultas contienen información relacionada, que aunque no sea exactamente lo que el usuario está buscando, puede ser de su interés. Esta filosofía de interacción con el usuario se aplicó con éxito en [Barranco et al., 2004] a la búsqueda inmobiliaria.

Dado que los tipos de datos difusos que vamos a emplear sólo van a encontrarse como rango de propiedades de tipo de dato `owl:DatatypeProperty`, la transformación que hemos visto en la apartado 3.3.1 solo tiene que representarlos como atributos en la relación correspondiente en el esquema de individuos. Si la ontología de dominio empleada para generar el esquema de base de datos no contiene instancias, entonces únicamente debemos indicar como rango de las propiedades de objeto un nuevo tipo definido por nosotros que representa atributos con datos difusos.

Si la ontología inicial contiene instancias es necesario proporcionar mecanismos para representar las instancias de los datos difusos en esa misma ontología. Dado que las ontologías OWL no soportan tipos de datos difusos es necesario extenderlas. A estas ontologías extendidas las llamamos ontologías OWL Like. Crearemos un nuevo tipo de XML Schema denominado *NumericFuzzyData*. Este tipo de dato soporta valores del tipo *CrispValue* para valores precisos, *TrapezoidalValue* para distribuciones de posibilidad trapezoidal, *IntervalValue* para valores de rango, *GreaterThanValue*, *LessThanValue* para valores de umbral superior e inferior respectivamente y *ApproximateValue* para distribuciones de posibilidad triangular. Estos tipos difusos soportados se muestran en la Figura 3.3.

La introducción de este nuevo tipo de dato de XML Schema provoca que estas nuevas ontologías no sean estándar. Por tanto, todas las tareas de razonamiento realizadas por los razonadores estándar no podrán llevarse a cabo. En lugar de crear un nuevo razonador desde cero que sea capaz de trabajar con estos tipos de datos, utilizaremos las capacidades que nos proporciona el SGBDOR subyacente. Esto nos permitirá realizar tareas de razonamiento sencillas en la propia base de datos donde se encuentra almacenada toda la información.

3.4. Ejemplo de Almacenamiento de Ontologías

En esta sección presentaremos un breve ejemplo para ilustrar mejor la forma en que se almacenan los datos en el esquema propuesto. Utilizaremos la conocida ontología de vinos

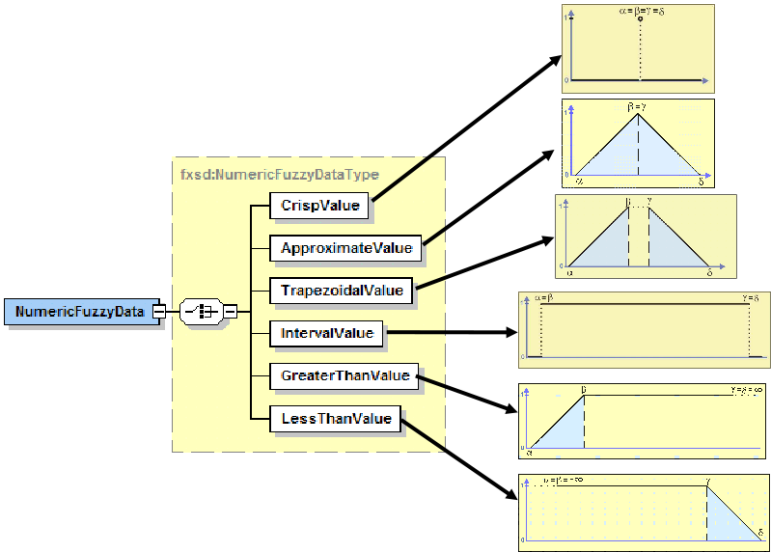


Figura 3.3: Tipos de Datos Difusos

Wine Ontology. En este ejemplo mostraremos la forma en que representamos la ontología en el modelo híbrido usando los principales elementos que se modelan de OWL. Dado que esta ontología es extensa, nos limitaremos a mostrar únicamente un ejemplo por cada uno de los distintos elementos representados.

Antes de comenzar con la representación de la ontología OWL, nuestra estructura debe contener los elementos básicos que nos permiten definir la ontología. Así pues definimos clases como `owl:Thing` y `owl:Nothing` y las diferentes propiedades que vamos a utilizar tales como `rdfs:SubClassOf`, `owl:IntersectionOf`, etc...

En la Figura 3.4 podemos ver como se almacena la ontología de inicio, en la tabla `Ont_Ontologies` indicamos el nombre de la ontología y se le asigna un identificador, del mismo modo creamos una entrada para el espacio de nombres en la `Ont_Namespaces` indicando el prefijo y la URL. A continuación se almacenan los términos correspondientes a la ontología, y a las clases predefinidas, en cada caso se indica el tipo de elemento que es. Una vez almacenados los elementos básicos comenzamos a almacenar las propiedades, primero se almacenan como términos en su tabla correspondiente y luego se almacenan como propiedades de tipo de objeto en la tabla `Ont_ObjectProperties`. En esta tabla nos encargamos de indicar sus propiedades, transitividad, simetría, etc... e indicamos el dominio y el rango de la propiedad en base a los términos en `Ont_Term` que modelan las clases a las que hacen referencia.

Con estos elementos básicos de inicio ya podemos indicar construcciones complejas tales como que una clase X es subclase de otra Z, etc...

Una vez inicializada la estructura comenzaremos a incluir los elementos, comenzando por las clases. Una clase básica se traduce en una tabla en el esquema de individuos y en una entrada en la tabla `Ont_Terms` del esquema de metadatos de la ontología. En la Figura 3.5 vemos como la clase *Winery* se almacena como término y que se construye su

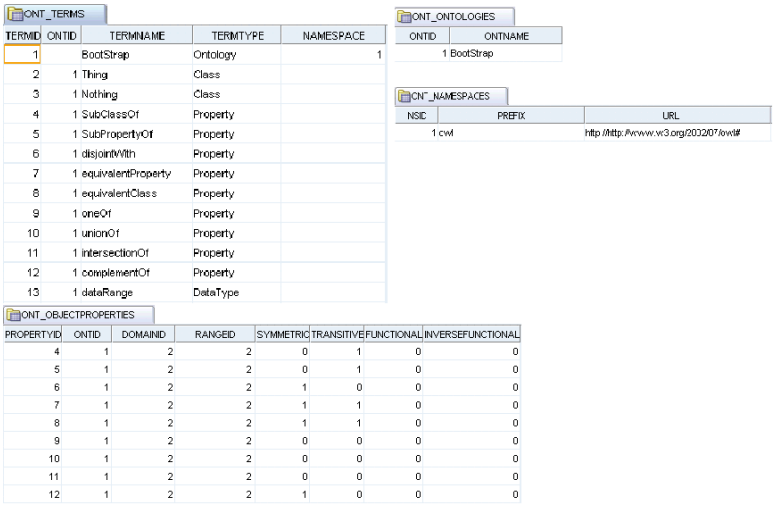


Figura 3.4: Almacenamiento de la ontología de inicio

tabla. Por otra parte, la clase *PotableLiquid* es subclase de *ConsumableThing*, por tanto, generamos las tablas para cada una, haciendo que el identificador de *PotableLiquid* sea clave externa del identificador de *ConsumableThing*. Desde el punto de vista del esquema de datos hemos reflejado la relación, pero también es necesario reflejarla en términos de la ontología, para ello, incluimos una entrada por cada clase en la tabla `Ont_terms` y una adicional en `Ont_Relationships` donde indicamos que existe una relación entre ambas, y esta relación viene dada por la propiedad `rdfs:SubClassOf` de la ontología de inicio.

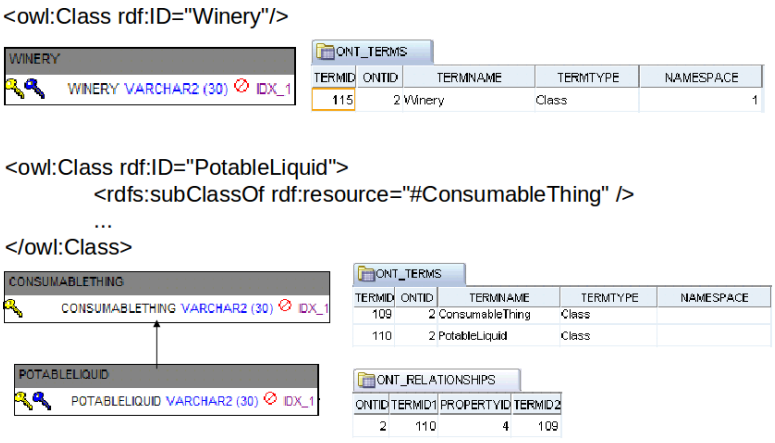


Figura 3.5: Almacenamiento de Clases

En la figura 3.6 podemos ver como se traducen las propiedades de objeto en el esquema. En el primer ejemplo tenemos una propiedad *madeFromGrape* que relaciona *Wine* y *WineGrape*. Dado que no establece ninguna relación de cardinalidad podemos entender que un vino puede estar hecho de distintas variedades de uva, y que a partir de un tipo de

uva se pueden hacer distintos tipos de vino. Por lo tanto, tenemos una relación n:n por lo que se crearía una tabla intermedia *madeFromGrape* con referencias a *Wine* y *WineGrape*. En cuanto a la estructura de la ontología almacenaríamos en la tabla `Ont_Terms` la nueva propiedad, que sería definida completamente en la tabla `Ont_ObjectProperties`.

En el segundo ejemplo creamos *hasColor* una subpropiedad de *hasWineDescriptor* de tal modo que creamos una nueva tabla que referencia a ésta. Además se incluye en `Ont_ObjectProperties` la nueva propiedad indicando los identificadores del dominio (el mismo que *hasWineDescriptor*). Dado que la propiedad es funcional, debemos indicarlo en la columna correspondiente.

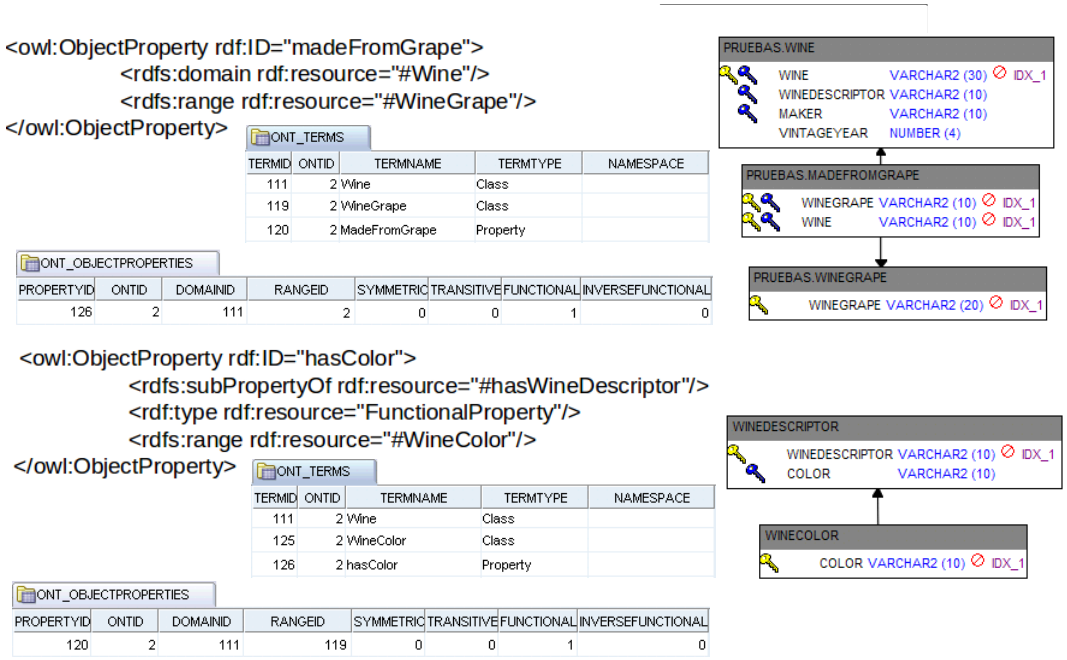


Figura 3.6: Almacenamiento de Propiedades de Objeto

En el caso de las propiedades de tipo de dato que se muestran en la Figura 3.7 vemos como para crear una propiedad, se añade un atributo a la tabla que representa su dominio, el tipo de este atributo será el indicado en el rango de la propiedad. En el ejemplo vemos como el proceso es completamente igual en el caso en que usamos tipos de XSD como cuando empleamos nuestro tipo especial para manejar datos difusos. En cuanto a la estructura de la ontología, es necesario indicar que existen nuevas propiedades en la tabla `Ont_Terms` y podemos describirla completamente en la tabla `Ont_DatatypeProperty`

Las restricciones en OWL son una forma especial de definir clases, definen una clase anónima con todos los individuos que cumplen esa restricción. Las restricciones pueden ser de valor o de cardinalidad. En ambos casos desde el punto de vista del esquema de individuos estas restricciones se modelan como vistas sobre los datos. En cuanto al componente estructural como se puede observar en la Figura 3.8, tenemos una tabla `Ont_Restrictions` donde podemos guardar la restricción y asignarle un nuevo identificador de clase.

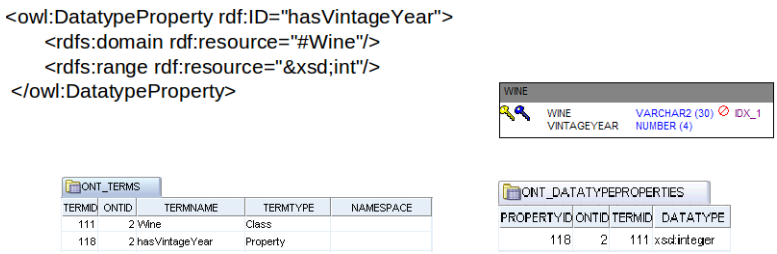


Figura 3.7: Almacenamiento de Propiedades de Tipo de Dato

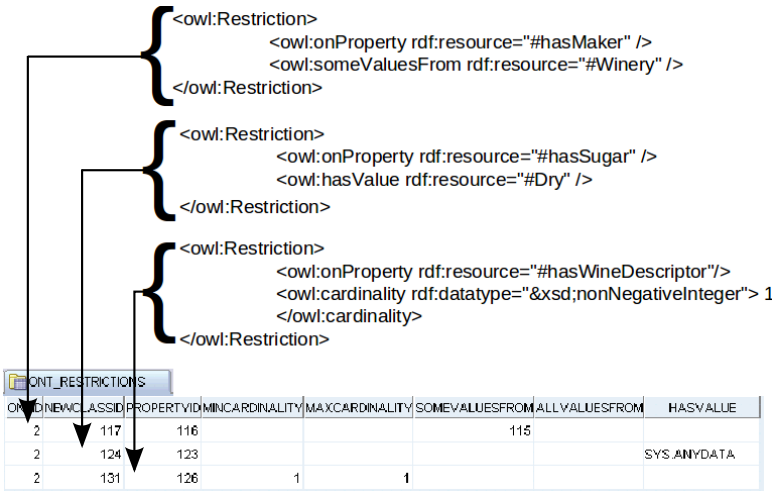


Figura 3.8: Almacenamiento de Restricciones

En la Figura 3.9 podemos ver como se modelan las descripciones de clase complejas, concretamente la correspondiente a la propiedad `owl:unionOf`. Aunque el proceso seguido para las propiedades `owl:intersectionOf` y `owl:complementOf` sería equivalente. En el esquema se crea una vista que es la unión de las tablas correspondientes a las clases a unir, y se almacena la lista de clases en `Ont_ClassList`, e introducimos el identificador de la lista, y de la nueva clase creada como unión de sus componentes en `Ont_Terms`. Una vez hecho esto, sólo queda incluir una relación en `Ont_Relationships` entre la clase creada y la lista de clases que la componen, en este caso la propiedad usada es `owl:unionOf` que era parte de la ontología de inicio.

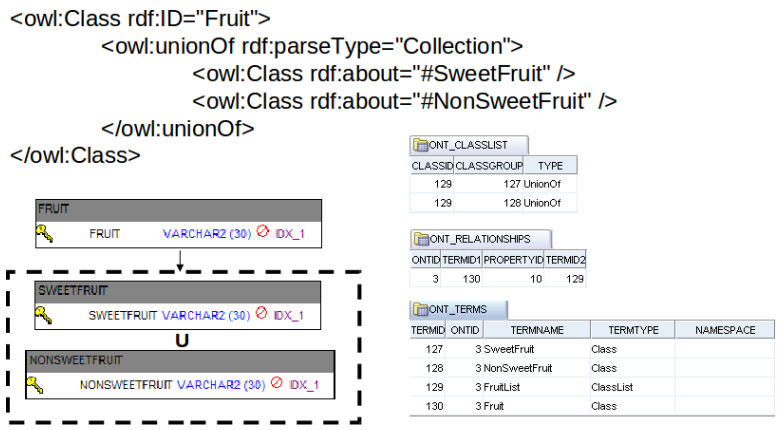


Figura 3.9: Almacenamiento de Clases Complejas. Unión de Clases

El ejemplo de la Figura 3.10 es bastante más complejo, ya que la propiedad `owl:oneOf` permite definir una clase en base a las instancias que puede tener. La representación de esta propiedad en un esquema de bases de datos se realizaría por medio de la implementación de un disparador. En cuanto al almacenamiento de la propiedad de la ontología como tal, primero debemos definir un término como grupo. Podemos definir los individuos que componen el grupo en `Ont_oneOf`, y creamos una relación donde indicamos que la clase *WineColor* debe de tener al menos uno de los elementos del grupo que acabamos de crear.

Finalmente en la Figura 3.11 podemos ver como se gestiona todo lo relacionado con la identidad de los individuos. Concretamente en este ejemplo mostramos como tratar la propiedad `owl:sameAs` aunque las propiedades `owl:differentFrom` y `owl:AllDifferent` se gestionan de forma análoga. Vemos con en la tabla `Wine` tenemos las instancias *MikesFavoriteWine* y *StGenevieveTexasWhite*, esta tabla estará asociada a su correspondiente término en la tabla `Ont_Terms` a través de una entrada en la tabla `Ont_ConceptMappings`. La propiedad nos dice que son iguales, por tanto debemos crear un grupo que los incluya en `Ont_IndividualsIdentity`. Cuando queramos comprobar la identidad de las instancias no tenemos más que mirar los grupos creados en la tabla `Ont_IndividualsIdentity`.

Como se puede ver aunque el proceso es algo complejo, se puede hacer de forma automática sin mucha dificultad.

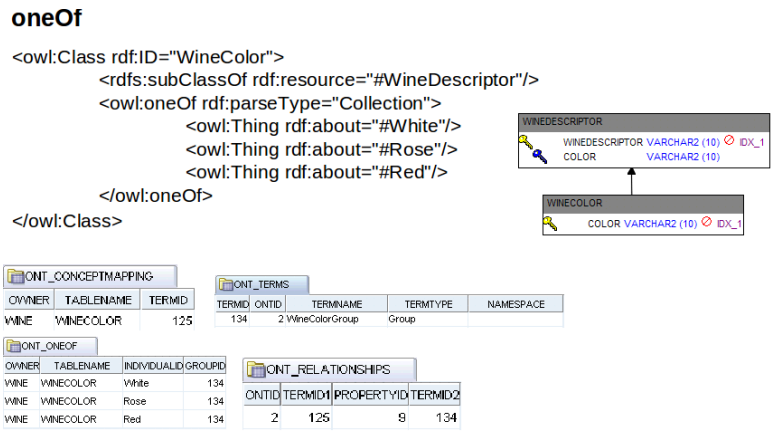


Figura 3.10: Almacenamiento de Clases Complejas con `owl:oneOf`

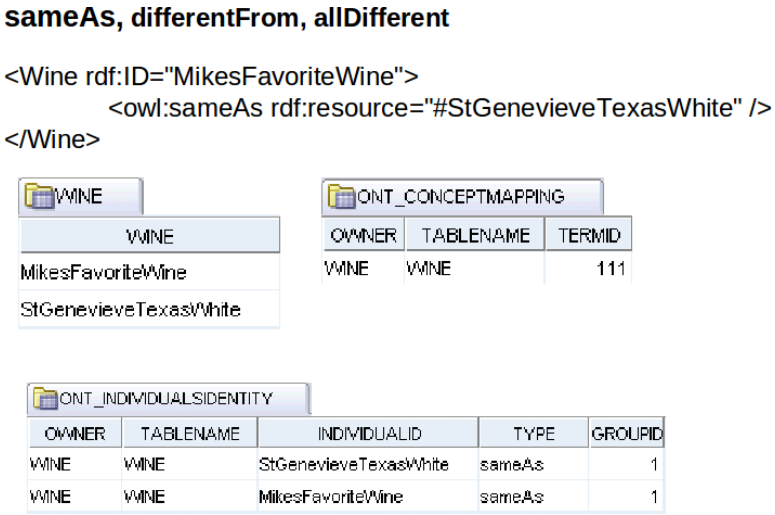


Figura 3.11: Esquema de Ontología e Individuos

3.5. Conclusiones

En este capítulo hemos presentado un algoritmo y un esquema de almacenamiento para ontologías OWL DL que permite usar ontologías de dominio como modelo conceptual para el diseño de esquemas de bases de datos. Esto permite crear el esquema desde cero con el objetivo de almacenar datos sobre instancias de una ontología en el contexto de la Web Semántica. El esquema y algoritmos propuestos cubren las principales estructuras de las ontologías OWL DL y preparan el terreno para desarrollar cierto grado de razonamiento básico en la base de datos.

El principal beneficio de esta aproximación es la obtención de esquemas sin el esfuerzo adicional de diseñar todo el modelo conceptual. Además de la posibilidad de consultar y manejar instancias de forma semántica. Las consultas sobre la ontología OWL se pueden traducir a consultas sobre la base de datos, aprovechando los mecanismos que ésta posee para el manejo de gran cantidad de datos de forma optimizada.

Los trabajos futuros se centrarán en la definición de procedimientos para realizar un razonamiento limitado, principalmente basado en la subsunción de conceptos. Aunque las propiedades empleadas son compatibles con OWL2 estamos trabajando en un versión que aproveche todas las capacidades de esta nueva versión del lenguaje. Un tema importante es el de la definición de restricciones complejas utilizando una representación adecuada basada en disparadores. También se debe tener en consideración todo lo referente a evolución de ontologías, un aspecto que por ahora no habíamos contemplado. Una vez finalizados todos estos aspectos, el objetivo final es la elaboración de una una interfaz gráfica para importar y exportar ontologías.

Capítulo 4

Representación Semántica de Textos No Estructurados

Actualmente las bases de datos relacionales son un componente esencial de los sistemas de gestión de información. Pese a ser una tecnología ampliamente aceptada e implantada, no ha dejado de evolucionar para adaptarse a las necesidades de información de cada momento. Así, la aparición de los modelos objeto-relacionales surgió como respuesta a la necesidad de gestionar información más compleja que la que se venía tratando hasta el momento. Los sistemas geográficos de información, por ejemplo, pusieron de manifiesto la necesidad de buscar nuevas fórmulas a la hora de gestionar información estructuralmente compleja. En la actualidad esta situación se está volviendo a repetir, el cambio de paradigma que ha introducido la aparición de la Web Semántica y la fuerza con que esta se está implantando, ha impulsado una serie de nuevas necesidades de información donde el objetivo no es únicamente la gestión de la información en bruto, sino también la obtención a partir de esta de una semántica que permita tratarla según su significado.

Actualmente la gestión y representación de la semántica de la información se realiza por medio de ontologías, así pues, es necesario realizar un tratamiento de la información desde un punto de vista ontológico. En el capítulo 3 hemos visto como es posible dotar a la estructura de la base de datos de cierta semántica, haciendo corresponder las estructuras de la base de datos con las de una ontología. Este pequeño paso puede aportar grandes beneficios, pero no resuelve por completo el problema, puesto que los datos contenidos en esas estructuras pueden tener su propia semántica.

Supongamos por ejemplo una base de datos bibliográfica, donde se almacena información acerca de autores, publicaciones, palabras clave de las publicaciones, etc... Supongamos que los distintos libros están agrupados en géneros y que estos están organizados en forma de jerarquía, donde por ejemplo dentro de *Ciencia-Ficción* tenemos *Ucronía*, *Distopía*, etc... a su vez estos pueden contener a otros, y así sucesivamente. Al tener asociada la estructura a una ontología tal y como se proponía en el capítulo 3, podríamos hacer una búsqueda en la que preguntásemos por libros de la categoría *Ciencia-Ficción*, con la ayuda de los mecanismos de subsunción podríamos determinar en la ontología la clase o clases que satisfacen la respuesta, y recuperar todas las instancias pertenecientes a dichas clases

y sus subclases.

Como puede apreciarse, este proceso nos permite añadir algo más de semántica a los datos que poseemos, podemos identificar que una instancia correspondiente a un libro es una *Distopía* y que por tanto es un libro de *Ciencia-Ficción*. Ahora supongamos que además del género, tenemos un pequeño texto junto a la entrada del título con una sinopsis que resume el contenido y describe en líneas generales el argumento. Un humano al leer este texto sería capaz de determinar sobre qué trata el libro, y obtener información adicional y relevante. Por supuesto es bastante complejo obtener información de un texto que no está estructurado, es una tarea que prácticamente sólo está al alcance de los humanos. Actualmente existe una gran cantidad de información textual almacenada en bases de datos de la cual no se está obteniendo el rendimiento suficiente. Por lo general sobre estos campos textuales sólo se suelen realizar búsquedas por palabras clave, todo desde un punto de vista meramente sintáctico.

Nuestra propuesta para este capítulo, es la de presentar una metodología para el procesamiento de los textos cortos no estructurados que aparecen en los campos textuales de las bases de datos, representando sus características más relevantes y tratando de obtener cierta semántica a partir de dichas características. El objetivo es el de obtener una ontología que represente el dominio de un atributo de la base de datos que contiene los textos cortos. Para ello primero deben de realizarse una serie de pasos, extraer la información relevante del texto por medio de la aplicación de técnicas de minería de datos y de textos, representar el texto como una forma intermedia provista de cierta estructura, y finalmente obtener la semántica de la estructura que se ha construido y a partir de ella obtener una ontología.

En nuestro caso particular partiremos de una columna de una base de datos que contiene textos cortos, procesaremos dichos textos y los representaremos mediante Conjuntos-AP. Recordemos que en la Sección 2.2.2 se realiza una introducción y una amplia descripción de este método. Una vez obtenidos estos Conjuntos-AP intentaremos dotarlos de contenido semántico empleando herramientas disponibles tales como bases de conocimiento y diccionarios electrónicos.

En las siguientes secciones veremos con más detalle cómo abordar cada uno de los pasos descritos anteriormente.

4.1. Metodología para la Representación Semántica de Textos no Estructurados

La metodología que proponemos es de carácter general, y permite la aplicación de una amplia variedad de herramientas para su implementación.

La propuesta consiste en un conjunto de etapas de procesamiento realizadas de forma secuencial, que permiten mediante el empleo de herramientas y métricas bien conocidas obtener información relevante sobre los campos textuales. Las etapas propuestas son las siguientes:

- **Preprocesamiento Sintáctico**: Esta primera etapa consiste en una limpieza de

carácter sintáctico donde se realiza la tokenización de los términos textuales, con la consiguiente eliminación de signos gramaticales y de puntuación. Es es esta etapa donde se realiza la eliminación de palabras vacías y, en caso de considerarse necesario, la lematización.

- **Preprocesamiento Semántico**: En esta etapa se realiza la detección de términos sinónimos, se determinan conjuntos de sinónimos, y se selecciona un término de entre ese conjunto que se considerará el representante canónico de dicho conjunto. El término representante del conjunto sustituirá al resto de términos en el texto a procesar.

  Para poder realizar la detección de sinónimos es necesario realizar las siguientes tareas:

  - *Etiquetado de categoría gramatical (Part Of Speech Tagging)*: Se etiqueta el término con la categoría gramatical que desempeña en la sentencia de la que se ha extraído. Esto es relevante de cara la la siguiente etapa.
  - *Desambiguación (Word Sense Disambiguation)*: Se determina el significado del término en cuestión, ya que, una vez conocido su significado, pueden determinarse sus sinónimos.
  - *Generación de conjuntos de sinónimos*: Los términos se agrupan en base a su significado obteniendo conjuntos de sinónimos. Durante esta etapa se puede hacer uso de diccionarios electrónicos para enriquecer los conjuntos de sinónimos con términos adicionales que no aparecen en el texto original.
  - *Selección del representante canónico de los conjuntos de sinónimos*: Para cada conjunto de sinónimos se determina su representante canónico, es decir el término que va a representar a todo el conjunto de sinónimos.

- **Representación Intermedia**: Una vez procesados los textos se procede a la representación de los rasgos más descriptivos del texto mediante una forma de representación intermedia. Sería deseable que esta forma intermedia aportase información adicional a la proporcionada por los términos base, y que permitiese una fácil traslación a una estructura ontológica.

- **Extensión Semántica**: Partiendo de la forma de representación intermedia, se buscan relaciones entre los términos contenidos en ella (teniendo en cuenta la información adicional que pueda aportar la propia estructura), mediante el empleo de diccionarios electrónicos o fuentes de conocimiento externas.

El proceso general de la representación de textos no estructurados queda reflejado de forma gráfica en la Figura 4.1, en la que podemos ver las distintas etapas de la metodología desde que se tiene el texto no estructurado, hasta que se genera la ontología de salida. Esta propuesta es genérica y por tanto se pueden utilizar diversas herramientas externas.

A continuación describiremos de forma detallada cada una de las distintas etapas indicadas anteriormente, para continuar haciendo un análisis pormenorizado atendiendo a las herramientas empleadas en cada etapa concreta.

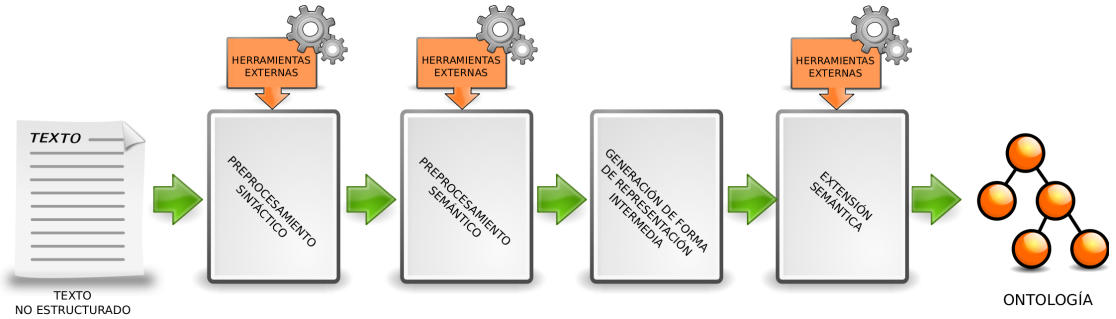


Figura 4.1: Proceso general para la representación semántica de textos no estructurados

4.2. Preprocesamiento de los Datos

El primer paso antes de abordar un proceso de minería de datos o de texto, es la preparación de los datos para obtener mejores resultados y facilitar el procesamiento. En este caso el procesamiento que hemos realizado tiene dos vertientes, una sintáctica y otra semántica.

4.2.1. Preprocesamiento Sintáctico

El preprocesamiento sintáctico se realiza con la herramienta de selección y filtrado descrita en el Apéndice B.3. Para el preprocesamiento semántico utilizaremos WordNet como diccionario externo (podemos encontrar una descripción de WordNet en el Apéndice A.5).

La herramienta de filtrado nos permite realizar un procesamiento sintáctico de los textos, a los que aplicaremos filtros para la tokenización y eliminación de palabras vacías (*stop-words*). Mediante este proceso eliminamos elementos sintácticos que no aportan información y eliminamos elementos de puntuación que complican el procesamiento automático del texto.

En este punto es necesario resaltar que no se considera conveniente realizar un preprocesamiento de las palabras mediante la aplicación de filtros de lematización tales como el de Porter [Porter, 1980], porque las formas lematizadas que ofrece como salida el algoritmo pueden no ser reconocidas como formas base por los diccionarios externos empleados, lo cual dificulta enormemente la tarea de preprocesamiento semántico.

Existen una amplia variedad de herramientas para realizar preprocesamiento sintáctico de datos textuales. Entre las herramientas más completas y más conocidas nos encontramos con:

- **ANNIE (a Nearly-New Information Extraction System)**: Es un sistema de extracción de información desarrollado por Hamish Cunningham, Valentin Tablan, Diana Maynard, Kalina Bontcheva, Marin Dimitrov y otros. ANNIE se presenta como un módulo que forma parte de GATE (General Architecture for Text Engineering) [Cunningham, 2002]. GATE (gate.ac.uk) es un proyecto de código abierto que proporciona soluciones para problemas de procesamiento de textos. ANNIE proporciona

herramientas para tokenización, detección de entidades (*Gazetteer*), división de sentencias (incluyendo una versión que utiliza expresiones regulares), un etiquetador de categoría gramatical, etc...

- **Lucene** : Proyecto de software libre implementado enteramente en Java, consistente en el desarrollo de una API para recuperación de información. La tarea básica de Lucene consiste en la indexación y recuperación de documentos, pero de forma adicional contiene algunos módulos con utilidades para realizar preprocesamiento de textos. Mediante el uso de las clases derivadas de *Analyzer* realiza tokenización, eliminación de palabras vacías y detección de frases para una amplia variedad de idiomas. También es capaz de realizar lematización utilizando una interfaz con el lenguaje de creación de algoritmos de lematización Snowball.

- **PrePro2010**: Es una herramienta de preprocesado de texto desarrollada por Ulli Waltinger de la Universidad de Bielefeld (Alemania), la cual permite la realización de tareas básicas de preprocesamiento tales como tokenización, detección de frases, lematización, etiquetado de categoría gramatical, detección de entidades, análisis de la estructura del documento y detección del lenguaje.

Esta etapa de la metodología podría realizarse con cualquiera de las herramientas citadas sin producir cambios significativos en el resultado final.

Una vez realizado el preprocesamiento desde el punto de vista sintáctico, los datos ya están preparados para trabajar sobre ellos. El siguiente paso consistirá en la generación de las estructuras de representación intermedia para los textos. Atendiendo al método de generación usado será conveniente realizar algún tipo de procesamiento adicional de los datos.

En el caso que nos ocupa, conviene recordar que la técnica que vamos a utilizar para representar los textos, se basa en el algoritmo Apriori [Agrawal and R.Srikant, 1994], consistente en la detección de itemsets frecuentes. En este caso los itemsets estarán compuestos por palabras o conjuntos de palabras. El problema reside en que un mismo concepto semántico puede estar representado por una amplia variedad de formas sintácticas o palabras. Esta dispersión puede provocar un efecto no deseado por el cual un concepto frecuente y relevante que sería detectado y representado mediante un Conjunto-AP, no lo sea al poseer una amplia gama de representaciones sintácticas. Para evitar esta situación nada deseable realizaremos un preprocesamiento semántico.

4.2.2. Preprocesamiento Semántico

El objetivo del preprocesamiento que vamos a llevar a cabo, es el de homogeneizar la representación sintáctica de los conceptos presentes en el texto. Lo que vamos a hacer es sustituir todas las palabras que son sinónimos de un concepto por una única forma sintáctica que actuará como representante canónico del conjunto.

Una primera aproximación al problema podría ser la de realizar un diccionario de términos sinónimos, sin embargo, dada la generalidad del dominio tratado esta tarea no es

abordable. Es por esto que parece más sensato acudir a diccionarios y bases de conocimiento ya existentes.

En este apartado vamos a definir de forma general el proceso para el incremento de frecuencias en términos sinónimos y posteriormente veremos como aplicarlo utilizando los diccionarios electrónicos disponibles.

Definición 1. *Sea $\mathcal{T} = \{T_1, \ldots, T_n\}$ un conjunto de textos. Dado un texto $T_i \in \mathcal{T}$, éste está compuesto por uno o más términos $t_j, j = 1, \ldots, n$. Cada término t_j tiene asociado uno o más significados $s_k, k = 1, \ldots, n$ dependiendo de los contextos en que se pueda emplear el término. Dado un término t_j cuyo contexto está determinado, este solo podrá tener un único significado desambiguado m_k. Un significado desambiaguado m_k puede ser el significado de uno o más términos t_j. Así, llamamos conjunto de sinónimos S_k a todos aquellos términos t_j cuyo significado desambiguado sea m_k.*

Definición 2. *Para cada conjunto de sinónimos S_k existe un término $t_j \in S_k$ que recibe el nombre de representante canónico de S_k, que es único y se denota como r_k. Todo término t_j perteneciente al conjunto de sinónimos S_k, puede ser sustituido por el representante canónico del conjunto r_k, sin que cambie la semántica del texto original.*

$$\forall t_j \in S_k, \exists r_k | t_j = r_k \tag{4.1}$$

Determinar el conjunto de términos sinónimos a un término dado es una tarea compleja, que a su vez incluye la realización de otras tareas que a su vez entrañan cierta dificultad. Para obtener los sinónimos de un término debemos conocer el significado de todos los términos implicados en la operación. Para poder determinar el significado de un término debemos conocer su categoría gramatical, ya que términos con la misma forma sintáctica tienen significados diferentes según su forma gramatical. Una vez determinada la forma gramatical y el contexto en el que se usa, podemos determinar el significado de un término, y con él, el conjunto de sus sinónimos.

Veamos con mas detalle en que consiste cada uno de estos procesos.

Etiquetado de Categoría Gramatical

El etiquetado de categoría gramatical consiste en añadir a cada término de un texto una etiqueta indicando la categoría gramatical desempeñada por dicho término en el contexto de una frase, ya sea *nombre*, *verbo*, *adjetivo*, *adverbio*, etc... . La determinación de la categoría gramatical correspondiente a un término puede realizarse en base a la definición del término, o en base al contexto en que este se emplea.

El proceso de etiquetado es bastante complejo, ya que todo él depende del contexto en el que se encuentre el término a etiquetar. Un mismo término puede tener distintas categorías gramaticales según el contexto en el que se use. Tomemos como ejemplo el término '*vino*' según el contexto en el que se use puede ser un nombre *"El joven bebe vino"*, o un verbo conjugado *"¿Cuando vino el joven?"*.

En la bibliografía existen diferentes aproximaciones a la hora de abordar el problema del etiquetado gramatical, en general éstas se pueden dividir en dos grandes grupos:

- **Aproximaciones Lingüísticas**: Se basan en la construcción de conjuntos de reglas por parte de expertos lingüistas para determinar la categoría de cada palabra atendiendo a una serie de patrones.

- **Aproximaciones Basadas en Corpus**: El conjunto de aproximaciones empleadas en este apartado se basan en técnicas de aprendizaje sobre grandes conjuntos de datos. El aprendizaje puede realizarse de modo supervisado o sin supervisión. En modo supervisado se tiene un corpus etiquetado y a partir de él se construye un clasificador que inferirá las etiquetas para los nuevos términos introducidos. Por el contrario, en el modo no supervisado se aprende a partir de un corpus sin etiquetar, teniendo en cuenta únicamente valores estadísticos de los datos.

Existen además aproximaciones híbridas que combinan las técnicas anteriores para obtener mejores resultados.

A continuación mostramos algunos de los etiquetadores más utilizados. Todos ellos son etiquetadores estadísticos, ya que son los que ofrecen mejores resultados hasta el momento.

- **TreeTagger** [Schmid, 1999] : Etiquetador basado en modelos ocultos de Markov (HMM), anota los textos con información acerca de la categoría gramatical y el lema del término correspondiente. Desarrollado en el seno del proyecto TC en el Instituto de Lingüística Computacional de la Universidad de Stuttgart. Ha sido utilizado con éxito para etiquetar textos en alemán, inglés, francés, italiano, español, griego, y es fácilmente adaptable a otros lenguajes si se dispone de un *lexicon* y de un corpus marcado manualmente.

- **TnT** [Brants, 2000]: TnT, es la forma breve de *Trigrams'n'Tags*, este etiquetador también se basa en modelos ocultos de Markov y su uso está muy extendido. Desarrollado en la Universidad des Saarlandes, TnT es muy eficiente y es posible entrenarlo para diferentes lenguajes y virtualmente cualquier conjunto de etiquetas. Se entrena mediante un corpus anotado y el sistema es capaz de tratar con palabras desconocidas.

- **SVMTool** [Giménez and Márquez, 2004]: Desarrollado en la Universidad Politécnica de Cataluña, es un etiquetador de código abierto que emplea Support Vector Machines (SVM) para clasificación.

- **Stanford Tagger** [Toutanova et al., 2003]: Desarrollado en la Universidad de Stanford, actualmente este etiquetador es el que mayor precisión obtiene para el idioma inglés.

Al etiquetar las palabras con su categoría gramatical estamos eliminando todos los sentidos pertenecientes a otra categorías gramaticales a la hora de realizar la desambiguación, lo que facilita la tarea de encontrar el sentido adecuado.

Desambiguación

Una vez etiquetadas las palabras de la entrada se procede a realizar la desambiguación. Esta parte es muy importante, puesto que depende de una buena desambiguación el que se identifiquen correctamente los sentidos de las palabras y que, por tanto, se elijan los conjuntos de sinónimos adecuados para representarlas.

Un algoritmo trivial de desambiguación, empleado en ocasiones como *benchmark* en evaluaciones de algoritmos de desambiguación, es el del sentido más frecuente. Dada una palabra y su categoría gramatical se toma como sentido aquel que es más habitual que tenga la palabra. Algunos diccionarios electrónicos almacenan valores de uso de los sentidos y suelen ofrecer los resultados ordenados de mayor a menor frecuencia de uso, por lo que nos bastaría con tomar como correcto el primer sentido que se nos ofreciese. Esta heurística no es suficiente para obtener resultados apropiados a nuestras necesidades, por lo que vamos a describir algunos algoritmos de desambiguación.

Comenzamos con un algoritmo básico en desambiguación basada en conocimiento, el algoritmo de Lesk. El algoritmo de Lesk [Lesk, 1986] es uno de los primeros algoritmos desarrollados para desambiguación semántica de todas las palabras en un texto sin restricciones. Lo único que necesita el algoritmo es un conjunto de entradas de diccionario, una por cada posible sentido de la palabra y conocimiento sobre el contexto inmediato en el que se realiza la desambiguación.

Aunque tradicionalmente se ha considerado un método basado en diccionario, la idea subyacente en el algoritmo de Lesk es la semilla para los algoritmos basados en corpus actuales. Casi todos los sistemas de desambiguación se basan de una forma u otra en alguna forma de solapamiento contextual, estando el solapamiento medido entre el contexto de una palabra ambigua y los contextos específicos correspondientes a varios significados de esa palabra, aprendidos de datos previamente anotados. La idea principal tras la definición original del algoritmo es desambiguar las palabras encontrando el solapamiento entre las definiciones de sus distintos sentidos. Dadas dos palabras W_1 y W_2 cada una con N_{W_1} y N_{W_2} sentidos definidos en un diccionario, por cada posible par W_1^i y W_2^j , $i = 1, \ldots, N_{W_1}$, $j = 1, \ldots, N_{W_2}$, primero se determina el solapamiento de las definiciones correspondientes contando el número de palabras que tienen en común. Luego, se selecciona el par de sentidos con el mayor valor de solapamiento y se asigna un sentido a cada palabra en el par inicial. A continuación se muestran los principales pasos del algoritmo.

```
(1) por cada sentido i de W1
(2)     por cada sentido j de W2
(3)         calcular Solapamiento(i,j), el número de palabras en común
            entre las definiciones del sentido i y el sentido j
(4) encontrar i y j para las que Solapamiento(i,j) es máximo
(5) asignar el sentido i a W1 y el sentido j a W2
```

El problema con este algoritmo es que cuando se intenta desambiguar más de dos palabras puede producirse una explosión combinatoria. Debido a esto, hay algunas variantes del algoritmo que intentan obtener soluciones mediante algoritmos de optimización tales como enfriamiento simulado.

Otra versión del algoritmo de Lesk, que también intenta resolver el problema de la explosión combinatoria de sentidos, es una variación simplificada que realiza un proceso de desambiguación distinto por cada palabra del texto de entrada. En este algoritmo simplificado de Lesk [Vasilescu et al., 2004] el significado correcto de cada palabra en un texto es determinado de forma individual, encontrando el sentido que produce un mayor solapamiento entre su definición del diccionario y el contexto actual. En definitiva se trata de buscar el sentido que contenga más palabras del contexto en el que se encuentra la palabra a desambiguar. En lugar de intentar encontrar el sentido a todas las palabras del texto de forma simultánea, se trata cada palabra individualmente. Los principales pasos de este algoritmo son:

```
(1) por cada sentido i de W
(2)     determinar Solapamiento(i), el número de palabras en común
        entre la definición del sentido i y el contexto actual de la frase
(3) encontrar el sentido i para el que Solapamiento(i) es máximo
(4) asignar el sentido de i a W
```

Este algoritmo simplificado mejora al original tanto en términos de precisión como de eficiencia. Es por esto que lo hemos tenido en cuenta como parte de los métodos de desambiguación que vamos a emplear.

Otra variante del algoritmo de Lesk, llamado algoritmo adaptado de Lesk fue introducido por Banerjee y Pedersen en [Banerjee and Pedersen, 2002]. En esta variante se emplean, además de las definiciones de las propias palabras, otras palabras relacionadas para determinar el sentido más probable para una palabra dado un determinado contexto. La novedad de esta aproximación es la información utilizada dado un sentido. Mientras que el algoritmo de Lesk original únicamente tiene en cuenta la definición de las palabras como contexto, el algoritmo adaptado utiliza las definiciones de conceptos relacionados. Por ejemplo, en el caso concreto de WordNet, el algoritmo tiene en cuenta la jerarquía de WordNet que define hiperónimos, hipónimos, holonimos, merónimos, tropónimos, relaciones de atributos y sus definiciones asociadas para construir un contexto más amplio para un significado de una palabra. Se extiende el contexto del diccionario con las definiciones de conceptos semánticamente relacionados. Este algoritmo mejora la precisión del algoritmo simplificado y por tanto también es un buen candidato a ser usado como método de desambiguación.

Selección del Representante Canónico

Una vez desambiguadas las palabras estas se asocian con el conjunto de sinónimos S_k correspondiente según el sentido que se les ha asignado. Obtenemos una serie de conjuntos de sinónimos $S_k, k = 1, \ldots, n$. Cada palabra o término t_j estará asociado a uno y sólo uno de los conjuntos S_k. En términos de semántica, todos los términos contenidos en el conjunto de sinónimos son equivalentes, por lo que realmente se podía escoger cualquiera de ellos de forma aleatoria. Sin embargo, consideramos más adecuado realizar la sustitución por el término mas frecuente en el texto original, ya que esto nos garantiza que el texto procesado seguirá siendo legible, evitando así en lo posible la introducción de términos no adecuados al contexto.

Por cada conjunto S_k tomamos los términos asociados a dicho conjunto y calculamos el número de veces que aparecen en el texto original. Ha de tenerse en cuenta que pueden aparecer términos cuya forma sintáctica sea idéntica, pero tengan distinta categoría gramatical, así pues debemos contabilizar únicamente aquellos términos asociados con el conjunto S_k que estamos procesando.

Una vez conocemos la frecuencia f de cada término t_j, seleccionamos aquel que aparezca más veces en el texto, el más frecuente, y lo designamos como representante canónico r_k. Una vez seleccionado el representante, se sustituyen todos los términos t_j del conjunto S_k por el término r_k.

4.3. Generación de Formas de Representación Intermedias y Extensión Semántica

Una vez realizado el preprocesamiento de los textos, se procede a la generación de la forma intermedia de representación. Esta representación puede ser cualquiera que pueda contener información acerca de los términos seleccionados y las relaciones existentes entre ellos.

Una vez determinada la forma de representación intermedia a utilizar, y expresado el conjunto de textos en dicha forma intermedia, se procede a la extensión semántica de la estructura. La extensión semántica de las estructuras creadas se realiza por medio del uso de herramientas externas que ya poseen información semántica. Una de las herramientas más utilizadas en estos casos es WordNet, aunque últimamente está acaparando mucha atención el uso del grafo de categorías de Wikipedia. Otras posibles fuentes para la extensión semántica podrían ser el *Open Directory Project*, o la ontología médica *Snomed*, en el caso de procesamiento de textos médicos. Cualquier ontología accesible de forma automática, es candidata para proporcionar semántica en la etapa de extensión.

Cualquiera que sea la herramienta externa seleccionada, ésta debe tener una estructura ontológica compuesta por conceptos y relaciones entre ellos. Lo mínimo que se puede exigir a estas estructuras es que contengan relaciones de tipo taxonómico, aunque cualquier otro tipo de relaciones adicionales también pueden ser de interés.

El procedimiento para la extensión semántica se compone de varias etapas. En la primera etapa deben buscarse correspondencias entre los términos extraídos del texto y los conceptos en la ontología/herramienta. Una vez establecida la correspondencia, se recorre la estructura desde los conceptos seleccionados hacia la raíz, y se van incluyendo nuevos conceptos encontrados en este camino. Los nuevos conceptos serán aceptados en base a diferentes criterios o métricas. El resultado final será una ontología que incluya todos los conceptos con los que se corresponden los textos inicialmente procesados, más aquellos que estén relacionados en la ontología/herramienta de referencia. En cierta forma podemos ver este proceso como una poda selectiva de una ontología general para adaptarla a un dominio específico.

4.4. Ejemplo de Aplicación de la Metodología

A continuación vamos a presentar un ejemplo de aplicación de la metodología empleando como forma de representación intermedia los Conjuntos-AP, y utilizando como recurso externo para el preprocesamiento semántico y la extensión semántica WordNet.

4.4.1. Representación Semántica de Textos No Estructurados

En apartados anteriores hemos presentado la metodología genérica para la transformación de textos en ontologías. En este apartado adaptaremos la metodología para el uso de la forma de representación intermedia de los Conjuntos-AP y de una herramienta ampliamente extendida como es el diccionario léxico WordNet [Fellbaum, 1998]. El uso de esta herramienta, puede ser útil para la realización de las tareas de procesado semántico y de extensión de las formas de representación intermedias.

La Figura 4.2 muestra la metodología general adaptada para el uso con WordNet, presentada originalmente en [Campaña et al., 2009], y que explicamos con mas detalle a continuación.

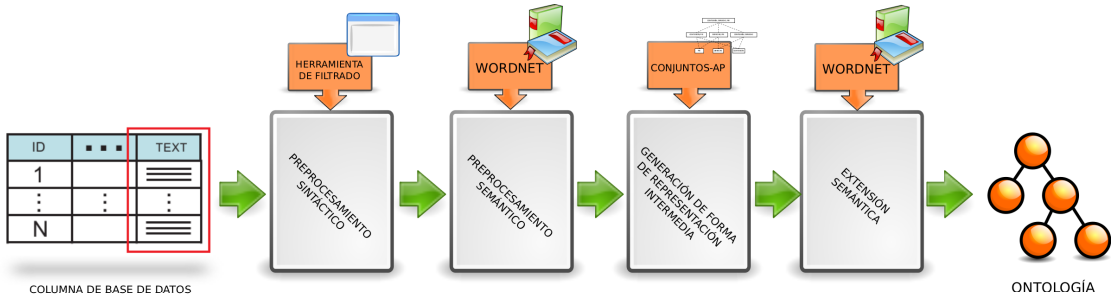


Figura 4.2: Metodología adaptada para el uso de WordNet

4.4.2. Preprocesamiento Semántico empleando WordNet

Como vimos anteriormente, un procesamiento semántico del conjunto de términos con el que vamos a trabajar nos permite adecuar su representación para el procesamiento específico que vamos a realizar. En este caso trataremos de sustituir cada término por su representante canónico en un conjunto de términos sinónimos. Para ello emplearemos el diccionario electrónico WordNet.

Recordemos que WordNet organiza sus términos en *synsets* o conjuntos de sinónimos, de modo que sólo debemos identificar el *synset* al que pertenece el término que debemos procesar y sustituirlo por el representante canónico del *synset*. Esta tarea, aunque a priori puede parecer bastante sencilla, en realidad se trata de un proceso bastante complejo que implica la resolución de otros problemas, que si bien han sido estudiados ampliamente, aún no disponen de soluciones completamente efectivas, y por tanto pueden ser una fuente de posibles errores.

Una palabra, independientemente de su forma sintáctica puede actuar como nombre, verbo, adjetivo, adverbio, etc... esto es a lo que se conoce como categoría gramatical (part-of-speech POS). Dependiendo de su categoría gramatical una palabra tendrá un sentido u otro. Es más, para una misma categoría gramatical una palabra puede tener varios sentidos. Cada uno de estos sentidos se corresponde con un *synset* que puede estar representado por varias palabras. Por tanto, nos encontramos con dos problemas, determinar la categoría gramatical de la palabra, y una vez obtenida esta, obtener el sentido correcto para la palabra. A esta última tarea se la conoce como desambiguación (Word Sense Disambiguation, WSD) y es un campo de investigación muy activo en el que trabajan multitud de investigadores por todo el mundo.

Etiquetado de Categoría Gramatical

Para resolver el problema de determinar la categoría gramatical de una palabra hemos recurrido a un programa de etiquetado gramatical desarrollado en la Universidad de Stanford basado en los trabajos publicados por *The Stanford Natural Language Processing Group* [Toutanova and Manning, 2000, Toutanova et al., 2003]. Este etiquetador, asigna una etiqueta a la palabra que indica su categoría gramatical. El conjunto de etiquetas utilizado por el etiquetador se corresponde con el conjunto de etiquetas *Penn Treebank English POS Tag Set* [Marcus et al., 1993]. WordNet únicamente utiliza cuatro categorías gramaticales, nombres, verbos, adjetivos y adverbios, por lo que es necesario adaptar el conjunto de etiquetas.

En la tabla 4.1 mostramos la descripción y correspondencia entre el conjunto de etiquetas *Penn Treebank* y el conjunto de etiquetas de WordNet, aquellas etiquetas que no tienen equivalente en WordNet se consideran palabras vacías y son eliminadas, puesto que no van a existir en WordNet.

Tras experimentar con este etiquetador hemos comprobado que cuando aparecen palabras poco comunes suele reconocerlas como nombres propios (NP o NPS), hemos encontrado casos en los que palabras técnicas o muy específicas han sido reconocidas de este modo. Esto es un problema, puesto que la categoría gramatical puede corresponderse con otra categoría y no necesariamente con la de nombre de WordNet (NOUN). En estos casos en los que nos encontramos con ausencia de información relevante, lo que vamos a hacer es probar a localizar la palabra en cualquiera de las categorías gramaticales, a la espera de encontrar una forma base adecuada.

Al etiquetar las palabras con su categoría gramatical estamos eliminando todos los sentidos pertenecientes a otra categorías gramaticales a la hora de realizar la desambiguación, lo que facilita la tarea de encontrar el sentido adecuado.

Desambiguación

Una vez determinada la categoría gramatical de cada una de las palabras del texto, estamos en condiciones de determinar su sentido. Para ello, de entre la multitud de métodos de desambiguación existentes nos hemos decidido por emplear el algoritmo adaptado de Lesk [Banerjee and Pedersen, 2002], dado que es un método pensado específicamente para

Tabla 4.1: Correspondencia entre Penn Treebank Tag Set y WordNet

Penn Treebank	**Descipción**	**WordNet**
CC	Coordinating conjunction	
CD	Cardinal number	
DT	Determiner	
EX	Existential there	
FW	Foreign word	
IN	Preposition or subordinating conjunction	
JJ	Adjective	ADJECTIVE
JJR	Adjective, comparative	ADJECTIVE
JJS	Adjective, superlative	ADJECTIVE
LS	List item marker	
MD	Modal	
NN	Noun, singular or mass	NOUN
NNS	Noun, plural	NOUN
NP	Proper noun, singular	
NPS	Proper noun, plural	
PDT	Predeterminer	
POS	Possessive ending	
PP	Personal pronoun	
PP$	Possessive pronoun	
RB	Adverb	ADVERB
RBR	Adverb, comparative	ADVERB
RBS	Adverb, superlative	ADVERB
RP	Particle	
SYM	Symbol	
TO	to	
UH	Interjection	
VB	Verb, base form	VERB
VBD	Verb, past tense	VERB
VBG	Verb, gerund or present participle	VERB
VBN	Verb, past participle	VERB
VBP	Verb, non-3rd person singular present	VERB
VBZ	Verb, 3rd person singular present	VERB
WDT	Wh-determiner	
WP	Wh-pronoun	
WP$	Possessive wh-pronoun	
WRB	Wh-adverb	ADVERB

WordNet y que aprovecha al máximo su estructura.

4.4.3. Generación de Formas Intermedias empleando Conjuntos-AP (AP-SETS)

Tal como vimos, a partir de una columna textual con ayuda del algoritmo Apriori se calculan los itemsets frecuentes del conjunto de términos extraído. Posteriormente se calcula la Estructura-AP de dominio y las Estructuras-AP inducidas para cada tupla.

Debido a la sustitución semántica sobre el texto realizada previamente a la generación de Conjuntos-AP, la frecuencia de ciertas palabras se ha visto aumentada de tal modo que conceptos representados por una amplia variedad de palabras no serán penalizados en el conteo de frecuencias.

Al construir los Conjuntos-AP obtenemos dos estructuras bien diferenciadas, por una parte se obtiene una Estructura-AP de dominio que describe en cierta forma el dominio de la columna de la base de datos que contiene los textos. Por otra parte, se obtiene una Estructura-AP por cada tupla de la relación original, esta Estructura-AP es la sub-Estructura-AP inducida por los términos textuales en la Estructura-AP de dominio y representa al texto corto.

En este caso nos vamos a centrar en la Estructura-AP de dominio, ya que esta contiene toda la información referente al dominio de la columna textual que se ha procesado.

4.4.4. Ejemplo de obtención de una Estructura-AP de dominio

A continuación se muestran unos breves ejemplos para facilitar la comprensión de los procesos descritos. Estos ejemplos han sido realizados sobre una selección de 62 títulos de artículos escritos por los miembros del grupo de investigación de *Bases de Datos y Sistemas de Información Inteligentes* de la *Universidad de Granada*, *España*.

Los parámetros de configuración del algoritmo Apriori empleados en el ejemplo son: N para el número de iteraciones, 0,03 para el umbral mínimo de soporte y 0,001 para el umbral mínimo de confianza.

Con esta configuración, la Estructura-AP obtenida es la que se muestra en la figura 4.3. En aras de la brevedad, la Estructura-AP está representada aquí por su itemsets maximales. Como se puede observar, la longitud máxima de los itemsets frecuentes encontrados es de 3 términos.

Debido a la naturaleza del proceso de generación, los itemsets carecen de orden interno. Trabajos futuros se ocuparán de la representación y la generación de un procedimiento, que tenga en cuenta el orden relativo dentro de los itemsets. Para superar este inconveniente, los términos son etiquetados con su *POS* con el fin de inducir un cierto orden, tal y como se verá más adelante. La etiqueta *POS* asignada a la palabra se muestra entre corchetes, en donde [A] significa adjetivo, [N] significa sustantivo, [V] significa verbo y [ADV] significa adverbio. Hemos seleccionado como conjunto de etiquetas POS de referencia, el conjunto de etiquetas empleado por WordNet, ya que para el tipo de procesamiento que vamos a realizar no es necesario emplear un repertorio de etiquetas más amplio, siendo suficiente con un conjunto más reducido y compacto.

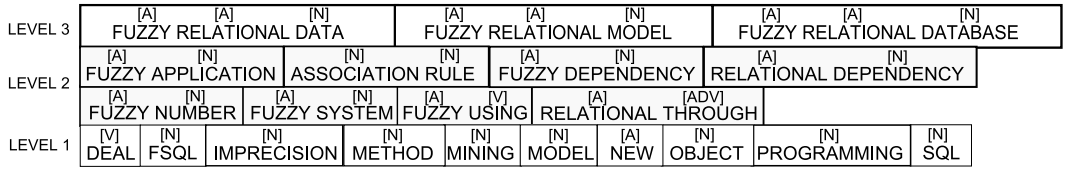


Figura 4.3: Itemsets maximales que definen la Estructura-AP

La Estructura-AP de dominio tiene una estructura de retículo, donde el primer nivel está dedicado a los términos individuales y niveles superiores contienen agrupaciones de términos frecuentes (el nivel 2 contiene pares, el nivel 3 triples, etc ...). Cada uno de los términos frecuentes de un itemset y las combinaciones entre ellos son también frecuentes.

El retículo completo se muestra en la Figura 4.4, debido al gran tamaño que éste presenta para ser visualizado lo resumimos en las Figuras 4.5 y 4.6. En las figuras podemos ver como aparecen coloreados los itemsets maximales, que cumplen la función de generadores del retículo. Para completar la estructura del retículo se incluyen dos nodos adicionales, el nodo vacío en la parte inferior de la figura y en la parte superior el nodo que contiene todos los términos que aparecen en el retículo.

Es interesante observar que el término `MODEL` aparece dos veces, y en ambos casos se ha determinado que su categoría gramatical es la de `Nombre`. Lo que en principio podía parecer un error, no es más que un efecto provocado por la realización del preprocesamiento semántico. Vemos que son dos términos diferentes, ya que si fuesen el mismo, `MODEL(2)` no sería un itemset maximal, sino que formaría parte del itemset maximal generador `FUZZY RELATIONAL MODEL`. Así pues, podemos comprobar que el preprocesamiento semántico ha asignado dos significados distintos en la desambiguación a `MODEL` y `MODEL(2)`. Atendiendo a las definiciones de WordNet nos encontramos que `MODEL` se refiere a *"Una descripción hipotética de una entidad o proceso complejo"*, mientras que a `MODEL(2)` se define como *"Simulación, representación de algo"*. Los contextos en los que usan estos términos han determinado un significado diferente para cada uno de ellos. Esta es unas de las razones por las que el preprocesamiento semántico es una etapa importante en la generación de los Conjuntos-AP, como ya se vió en la Sección 4.2.2.

Entre los itemset maximales representados en el retículo podemos observar algunos que ciertamente resultan chocantes, procederemos a analizarlos con el objetivo de determinar el porqué de su aparición y como debemos tratarlos. Lo primero que observamos es que algunos de los itemsets obtenidos, no aportan apenas información a pesar de ser muy frecuentes, esto ocurre principalmente con aquellos que contienen verbos y adjetivos. La razón de la alta frecuencia de estos elementos es la tendencia a una estructura repetitiva en los nombres de los artículos científicos. Así pues, por ejemplo, vemos que el término `NEW`, en este caso un adjetivo, no aporta ningún tipo de información pero es un elemento muy utilizado en los títulos para hacer especial hincapié en la novedad del método o aplicación propuesto en el artículo. Una situación similar se produce en el caso del verbo `DEAL`, es muy habitual encontrar títulos en los que se presentan métodos novedosos para tratar un problema específico conocido y que por tanto incluyen la expresión *"deal with"*. Debido a

la primera etapa de procesamiento sintáctico *with* se trata como una palabra vacía y se descarta, por lo que nos encontramos con la aparición aislada del término *deal*. Algo similar ocurre con `FUZZY USING` y `RELATIONAL THROUGH`, *use* y *through* son respectivamente un verbo y un adverbio muy utilizados a la hora de explicar de una forma compacta el contenido de un artículo mediante su título. En este caso al encontrarse la actividad del grupo de investigación centrada en la lógica difusa (*fuzzy logic, fuzzy numbers, fuzzy databases*) y en modelos de datos relacionales (*relational data, relational data model, relational database*), la estructura de los títulos y la actividad investigadora favorecen la aparición de estos términos, aunque son poco informativos. Así pues, vemos que la aparición de adjetivos, adverbios y verbos, si no están asociados a un nombre, en raras ocasiones aportan información relevante. Debido a este fenómeno, cuando procesamos el retículo debemos ser conscientes de este efecto y poder contrarrestarlo de alguna forma. Para ello, más adelante mostraremos un método sencillo que nos permite eliminar la terminología que no es adecuada desde un punto de vista semántico.

Con el objetivo de mejorar la legibilidad de los gráficos, vamos a utilizar un subconjunto del retículo completo donde se mostrará únicamente la parte más interesante de la estructura que se utilizará más tarde para ilustrar algunos ejemplos, además de omitir el nodo superior e inferior.

La Figura 4.7 muestra el retículo formado por los itemsets maximales `FUZZY RELATIONAL DATA`, `FUZZY RELATIONAL DATABASE`, `FUZZY RELATIONAL MODEL`, `SQL` y `FSQL`. Es importante recordar que el orden entre los términos en los itemsets no es tenido en cuenta por el procedimiento de generación. La Figura 4.7 muestra los itemsets ordenados para facilitar la comprensión. El problema del orden es tratado en un sección posterior. Aunque no se muestra en la figura es importante señalar que el retículo tiene un nivel inferior que contiene el conjunto vacío, y un nivel superior que es la unión de todos los itemsets maximales.

El retículo obtenido es la estructura que se empleará como punto de partida para la generación de la ontología. Para facilitar el seguimiento de los ejemplos, emplearemos la versión reducida del retículo que se muestra en la Figura 4.7.

4.4.5. Extensión Semántica de los Conjuntos-AP empleando Reglas Sintácticas

En el ejemplo que estamos siguiendo, la forma de representación intermedia que hemos seleccionado es la de los Conjuntos-AP, más concretamente la Estructura-AP de dominio obtenida de un campo textual de la base de datos. Esta Estructura-AP define un retículo. Un retículo se puede considerar como una ontología básica de inclusión. El retículo proporciona una estructura jerárquica básica que debe ser filtrada para ser de utilidad. En las siguientes secciones veremos como partiendo de una Estructura-AP, esta puede enriquecerse para generar una ontología que describa su contenido.

Nuestro primer enfoque considera el retículo como una ontología de inclusión. El problema es que contiene información no relevante para la construcción de la ontología y, además, el orden de los términos dentro de los itemsets no está fijado. Como se explicó con anterioridad, cuando la Estructura-AP se construye, el orden relativo entre los términos en los itemsets no se tiene en cuenta. Por lo tanto, la única información en la que podemos confiar

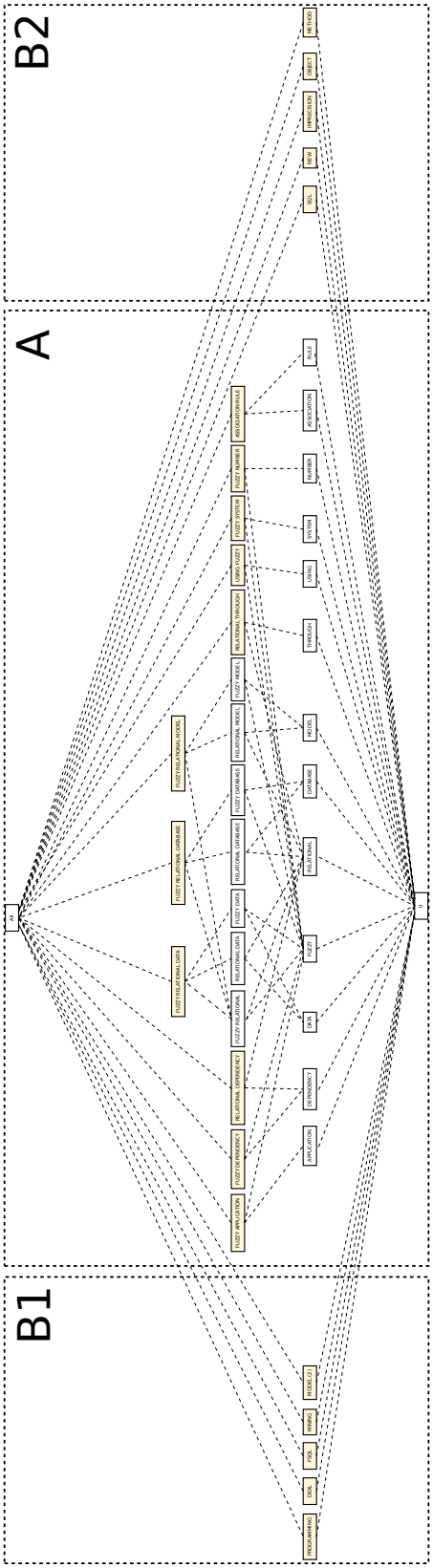


Figura 4.4: Retículo completo definido por la Estructura-AP

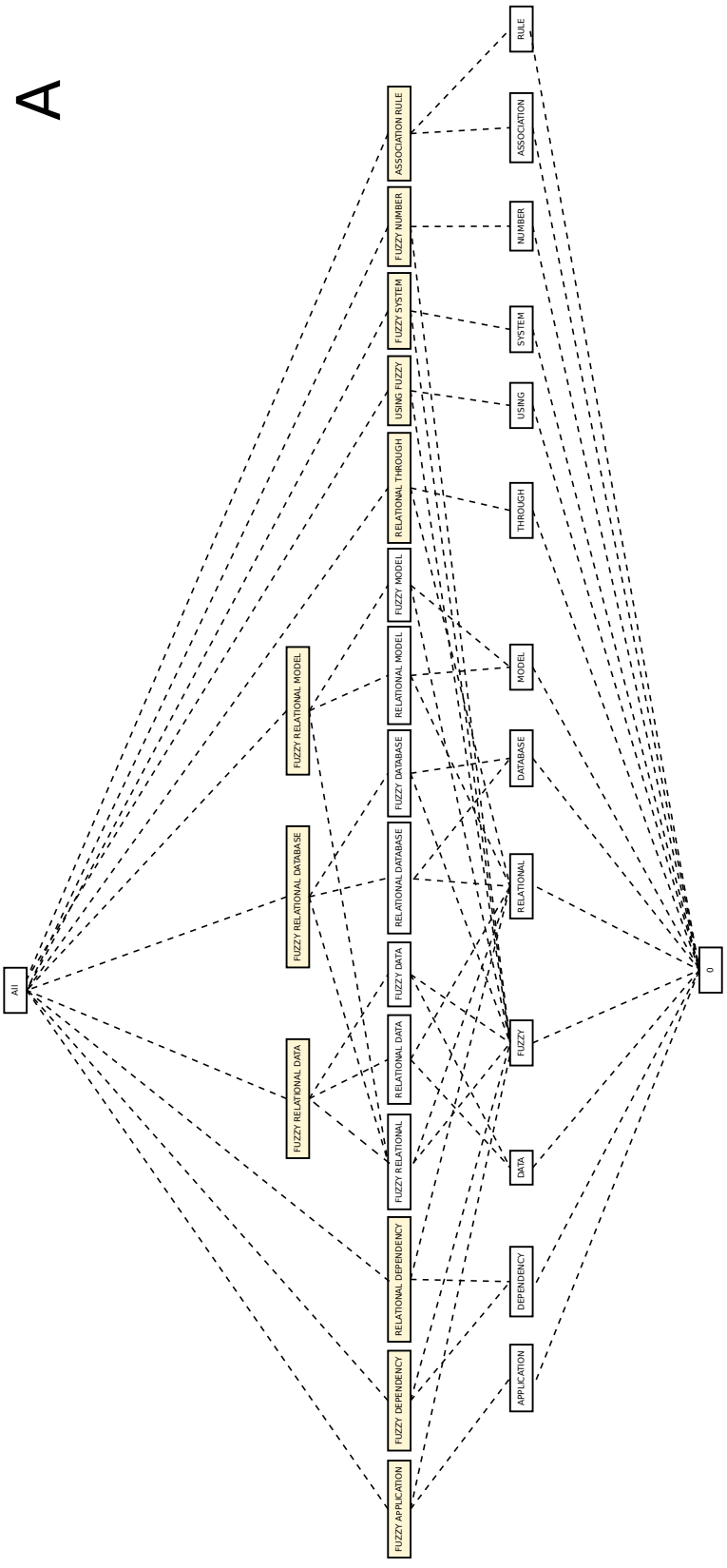


Figura 4.5: Retículo completo definido por la Estructura-AP

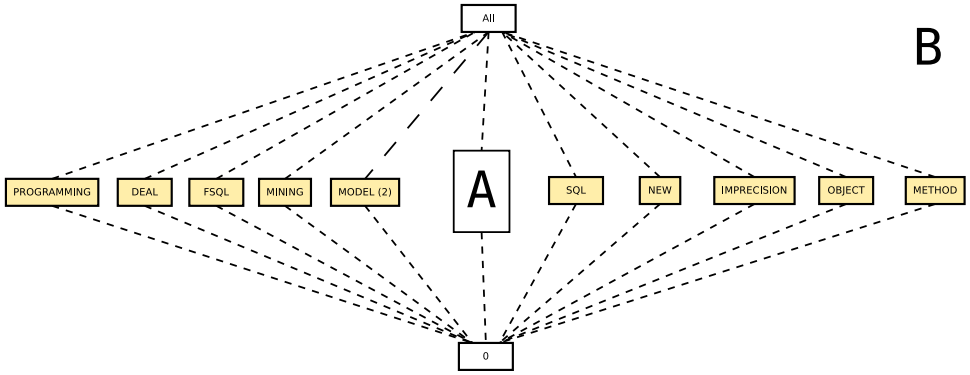


Figura 4.6: Retículo completo definido por la Estructura-AP

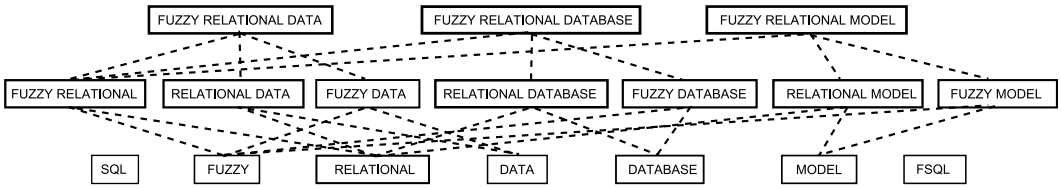


Figura 4.7: Fragmento seleccionado del retículo definido por la Estructura-AP

es que los términos que aparecen dentro de un itemset son frecuentes y suelen aparecer en el mismo contexto.

Podemos proporcionar un cierto orden a los términos en un itemset utilizando una serie de reglas sintácticas básicas. Estos itemsets ordenados pueden considerarse como candidatos para ser conceptos en la ontología.

El conjunto de normas aplicadas son las siguientes:

- **Nivel 1:** Los términos individuales son considerados candidatos a ser un concepto en la ontología si son nombres [N] o verbos [V].
- **Nivel 2:** Los itemsets compuestos por dos términos, se consideran candidatos a ser un concepto en la ontología si son nombres precedidos por un adjetivo [AN] o nombres compuestos [NN].
- **Nivel N:** Los itemsets candidatos son los que están compuestos por una combinación válida del nivel anterior, y están precedidos por un adjetivo o seguidos de un sustantivo. Por ejemplo, las combinaciones válidas para el nivel 3 son [AAN], [ANN] y [NNN].

Cuando un nodo del retículo no sigue estas reglas, el nodo se elimina. Una vez que todos los nodos son evaluados, el resto de nodos forman parte de una estructura conectada. El nodo en el nivel 0, que representa al conjunto vacío, se elimina.

Con el fin de transformar la estructura resultante de una ontología, cada término en el nivel 1 se hace un concepto subclase de la clase `owl:Thing`. Para cada nivel, los nodos

se incluyen como conceptos en la ontología y las relaciones entre itemsets se tratan como relaciones jerárquicas *IS-A*.

Siguiendo este enfoque se obtiene el fragmento de ontología mostrado en la Figura 4.8.

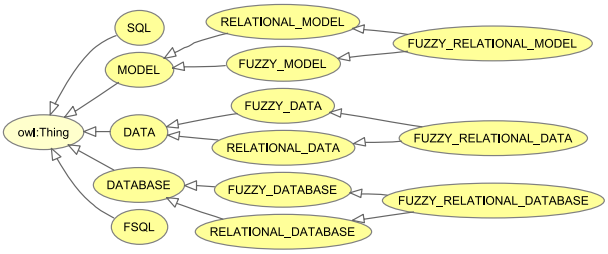


Figura 4.8: Fragmento de la ontología definido por la Estructura-AP

Como puede observarse en la figura, la ontología presenta una estructura razonable. Sin embargo, desde el punto de vista de la consulta esta ontología no añade ningún valor al proceso, ya que esta estructura es equivalente a una búsqueda realizada por palabras clave. No obstante, la ontología puede ser de gran utilidad para un usuario que no sabe nada sobre el contexto del campo de texto en el que va a buscar.

Dado que las palabras en los conceptos de la ontología son etiquetadas con su POS y se desambiguan, podemos incluir como propiedad de tipo de dato de una clase sus sinónimos, con el fin de enriquecer la búsqueda sintáctica. Esto aumenta las consultas con nuevos términos sintácticos permitiendo la búsqueda de un concepto utilizando las diferentes formas sintácticas que se utilizan para referirse a dicho concepto.

Este enfoque sólo tiene en cuenta propiedades y procedimientos matemáticos, y las normas sintácticas básicas. Para obtener alguna semántica de la estructura, en secciones posteriores proponemos otros enfoques basados en el uso de diccionarios léxicos y otros recursos semánticos.

4.4.6. Extensión Semántica de los Conjuntos-AP empleando WordNet

La Estructura-AP de dominio, tal y como se vio anteriormente en la sección 2.2.2 tiene una estructura de retículo, donde en el primer nivel se encuentran términos independientes, y conforme subimos de nivel obtenemos agrupaciones frecuentes de términos.

Un término de cardinalidad uno situado en el primer nivel, se corresponde con un posible concepto de la ontología, puesto que dicho término debido al procesamiento semántico realizado anteriormente es el representante canónico de un *synset* que define un concepto. Así pues, comenzaremos a construir la ontología generando una clase por cada término presente en el primer nivel de la Estructura-AP. Una vez tenemos las clases de partida vamos a enriquecer la ontología creando relaciones y nuevas clases.

Lo primero que vamos a hacer es generar una estructura jerárquica, a modo de taxonomía en la que iremos encajando los conceptos. Para ello, vamos a buscar relaciones de Hiperonimia/Hiponimia entre los términos del primer nivel. Por cada término buscamos todos sus hiperónimos, todos aquellos términos en los que esté contenido el significado del

synset. Cuando encontremos un hiperónimo común para dos o más términos crearemos una clase en la ontología para ese término. Esta clase se define como la clase padre de aquellas clases que la tienen como hiperónimo. Una vez terminado todo el proceso, aquellos conceptos que no tengan una superclase asociada, serán asignados como subclases de `owl:Thing`. Con esto obtendremos una ontología básica preliminar que enriqueceremos añadiendo las relaciones semánticas que podemos obtener de WordNet.

En la Tabla 4.2 mostramos la correspondencia entre algunas propiedades de las ontologías OWL y las relaciones semánticas que disponemos en WordNet.

Tabla 4.2: Correspondencia entre WordNet y OWL

OWL	**WordNet**
owl:Class	Synset
rdfs:subClassOf	Hipónimos/Hiperónimos
owl:inverseOf	Antónimos
owl:Thing	Superclase de todos los conceptos huerfanos

La ontología obtenida mediante la obtención de hiperónimos comunes no es lo suficientemente descriptiva, ya que debido a la naturaleza lingüística de WordNet, los hiperónimos comunes son términos abstractos generales. Así que no sólo incluimos su hiperónimo común, sino también toda la ruta entre las clases. Las clases son considerados como subclases de la última clase en la ruta obtenida. Aunque la hiperonimia y la hiponimia son relaciones léxicas y pueden considerarse equivalentes a relaciones de inclusión entre conceptos, utilizarlas de este modo para generar una estructura jerárquica es bastante útil.

Este enfoque tiene importantes desventajas. El uso de WordNet convierte la desambiguación en un punto clave en la generación de estructuras correctas. Una palabra mal desambiguada puede introducir la terminología y relaciones incorrectas en la ontología. Además, un inconveniente bastante importante es la falta de vocabulario técnico en WordNet. WordNet carece de mucha de la terminología de áreas de conocimiento específicas por lo que, en ocasiones, las ontologías generadas no son lo suficientemente relevantes. Los términos individuales que no se encuentran en WordNet se incluyen como subclases de `owl:Thing`, los términos compuestos se buscan en WordNet y se ignoran si no se encuentran.

La ontología obtenida mediante este método es muy grande y compleja. Debido a las limitaciones de espacio, la Figura 4.9 sólo muestra la rama que contiene los términos relacionados con los itemsets mostrados en el ejemplo de la Figura 4.8.

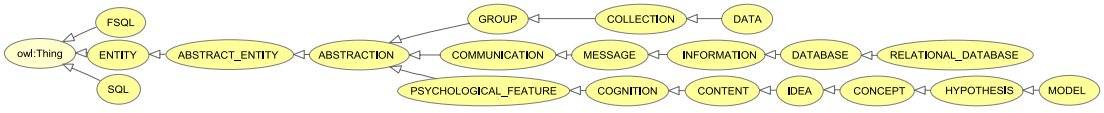


Figura 4.9: Fragmento de la ontología obtenido usando WordNet

Como puede verse, se pierden términos técnicos compuestos y vocabulario específico. El enfoque basado en reglas sintácticas ignora la semántica, aunque es capaz de generar una representación más adecuada de estas palabras. Este enfoque tiene la ventaja de proporcionar una terminología nueva y pertinente al contexto teniendo en cuenta la semántica, pero no cuando el vocabulario es muy específico o el proceso de desambiguación no es apropiado.

La combinación de ambos enfoques puede ofrecer un medio para incluir la terminología técnica desconocida, y la adición de significado semántico relevante a los conceptos y sus relaciones.

4.4.7. Extensión Semántica Combinada de los Conjuntos-AP empleando WordNet y Reglas Sintácticas

Este enfoque combinado trata de compensar las virtudes y los inconvenientes de los enfoques anteriores. La idea es poder tratar con terminología desconocida, mientras se da una semántica apropiada a la terminología conocida. El problema de desambiguación sigue ahí y la única solución es buscar una técnica de desambiguación apropiada.

Este enfoque comienza como el ejemplo básico empleando reglas sintácticas, primero se crea la ontología básica teniendo en cuenta las reglas sintácticas y una vez que la ontología básica está construida, se utiliza WordNet para buscar hiperónimos comunes a varios términos y sus caminos. El resultado es una ontología extendida que contiene la terminología desconocida correctamente colocada bajo los términos conocidos.

La Figura 4.10 muestra el mismo fragmento de la ontología de la Figura 4.9 pero obtenidos mediante el enfoque combinado.

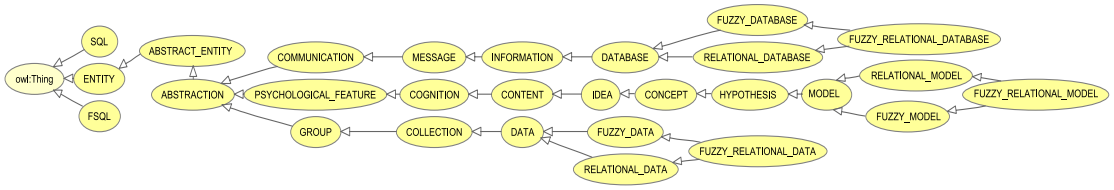


Figura 4.10: Fragmento de la ontología obtenida utilizando el enfoque combinado

Como puede verse, la terminología desconocida se conserva y se amplían los términos básicos utilizando WordNet.

La extracción automática de ontologías a partir de atributos de texto permite la descripción de los atributos textuales cuyo contenido es desconocido por el usuario. La ontologías generadas en la propuesta contienen una lista de términos de búsqueda para cada concepto que se define en la ontología. Estos términos se extraen de los sinónimos de WordNet, a partir del concepto modelado en la ontología. Los términos de búsqueda están codificados como propiedades de anotación, y puede ser utilizados para construir consultas sobre el atributo de texto.

Este enfoque presenta una ontología que se puede utilizar para mejorar las consultas proporcionando términos adicionales de búsqueda que permiten dotar de cierta semán-

tica al dominio compuesto por los atributos textuales procesados. Del mismo modo, la propia representación gráfica de la ontología, nos puede servir de guía a la hora de realizar consultas, puesto que cada uno de los conceptos representados lleva asociado multitud de términos de consulta.

4.5. Conclusiones

En este capítulo hemos introducido una metodología para la generación de ontologías a partir de textos no estructurados. La definición de la metodología es muy genérica y permite la implementación de cada una de sus etapas con diferentes herramientas. A modo de ejemplo hemos propuesto una aplicación de la ontología utilizando WordNet como soporte para el proceso de desambiguación semántica, los Conjuntos-AP como forma de representación intermedia y de nuevo WordNet como herramienta para la extensión semántica de la forma intermedia obtenida de los textos.

Como hemos podido comprobar por los resultados obtenidos en el ejemplo, el uso de WordNet como herramienta de extensión semántica no es del todo adecuado, debido principalmente a su orientación lingüística y a la falta de vocabulario técnico. Además a pesar de que existen diferentes versiones de WordNet y que éste se va actualizando, la inclusión e integración de nueva terminología no se sucede de forma habitual.

Así pues, sustituiremos el uso de WordNet en esta etapa por el uso de otro recurso cuyo uso se está extendiendo cada vez más, Wikipedia.

Capítulo 5

Representación Semántica de Textos No Estructurados usando Wikipedia

En la sección anterior hemos propuesto una metodología para la generación de ontologías a partir de textos no estructurados almacenados en una base de datos. Como ejemplo de implantación de la ontología se ha utilizado WordNet en la etapa de extensión semántica. Como hemos podido ver aunque WordNet es una herramienta muy útil para determinar sinónimos y realizar preprocesamiento semántico, su estructura está demasiado orientado a la lingüística y no proporciona los resultados que serían de esperar. En esta sección sustituiremos WordNet por Wikipedia en la etapa de extensión semántica. Realizaremos la extensión de los Conjuntos-AP mediante el uso de las categorías de Wikipedia.

Antes de comenzar con la generación de la ontología, debemos obtener la Estructura-AP. La generación de la Estructura-AP sigue el mismo proceso que se vio en el capítulo anterior. Recordemos de forma breve el proceso llevado a cabo:

- Inicialmente se determina un campo textual de la base de datos para procesar.
- Primero se preprocesan los datos aplicando los filtros correspondientes.
- Posteriormente se realiza la etapa de preprocesamiento semántico, con la finalidad de detectar términos sinónimos y aumentar las frecuencias de los conceptos correspondientes. Este proceso comprende las siguientes etapas:
  - Se etiquetan las palabras obtenidas con su categoría gramatical (POS Tagging).
  - Se realiza una desambiguación de las palabras teniendo en cuenta su categoría gramatical.
  - Se sustituye cada una de las palabras originales por el representante canónico del *synset* de WordNet al que pertenece.
- Una vez preprocesadas las palabras se obtiene la Estructura-AP de dominio que representa al campo procesado.

Utilizaremos la Estructura-AP obtenida como punto de partida para la generación de la ontología. Este proceso extendido con el uso de Wikipedia para la generación de ontologías se muestra en la Figura 5.1, donde se adapta el la metodología general para la extensión de ontologías para el caso particular del uso de WordNet en el preprocesado semántico, y Wikipedia en la etapa de extensión semántica de la ontología.

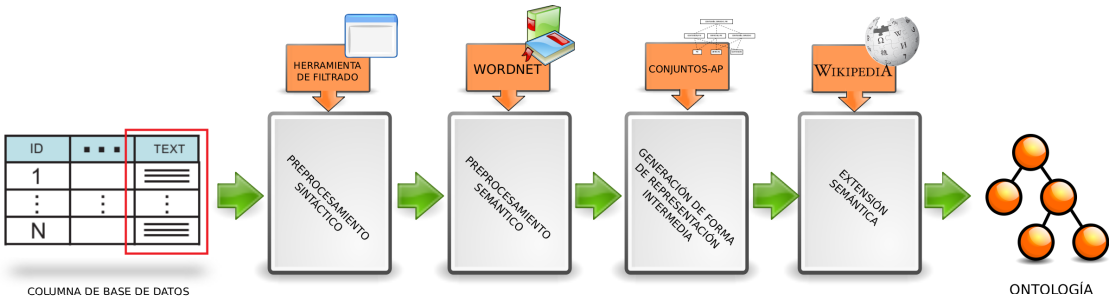


Figura 5.1: Proceso para la representación semántica de textos no estructurados basado en Wikipedia

En el capítulo anterior vimos como utilizar la estructura jerárquica de clases definida por WordNet para extender la ontología básica de inclusión que forma el retículo definido por la Estructura-AP. Dadas las particulares características de WordNet y su orientación lingüística, esta jerarquía contenía conceptos muy genéricos y abstractos, por lo que puede que ésta no sea la mejor elección como marco de referencia para la extensión de la Estructura-AP. Aunque los resultados obtenidos utilizando WordNet nos permiten extender la ontología, los nuevos términos no aportan toda la información que esperábamos.

Así pues, parece que la mejor opción para extender la Estructura-AP es partir de una ontología bien definida, concreta y diseñada desde un punto de vista práctico. Actualmente existen una gran cantidad de ontologías que presentan estas características, pero debido a su carácter dinámico y consensuado hemos decidido utilizar como ontología de referencia el grafo de categorías de Wikipedia.

En las siguientes secciones veremos algunos procedimientos para la selección de nodos del grafo para configurar un subconjunto conexo en forma de jerarquía que proporcione un contexto adecuado a los términos contenidos en la Estructura-AP.

5.1. El Grafo de Categorías de Wikipedia

Wikipedia es una enciclopedia en línea de carácter colaborativo en la que todos los usuarios pueden generar y editar contenidos. Cada una de las paginas de Wikipedia está asociada a una o más categorías. En las páginas de Wikipedia se define la categorización que se lleva a cabo, de la siguiente forma: *"La categorización es una herramienta provista por el software mediawiki que permite almacenar artículos y otras páginas dentro de categorías, que reúnen a varios artículos o páginas de características similares. Las categorías tienen a su vez subcategorías y supercategorías, permitiendo navegar de lo general a lo concreto y viceversa, a través de una estructura de árbol. Ayudan a los lectores a conocer qué artículos*

existen sobre un determinado tema, incluso sin saber de antemano si ya existen o con qué nombre aparecen.".

Por lo tanto tenemos un conjunto de categorías relacionadas entre sí mediante relaciones de hiponimia/hiperonimia. Cada una de estas categorías está definidas por los usuarios editores de Wikipedia, estos mismos son los que deciden las relaciones que va a tener cada una de estas categorías con el resto. Cada categoría puede tener un número arbitrario de subcategorías donde, por lo general, una subcategoría se determina por una relación de hiponimia o meronimia. Esto hace que el grafo de categorías de Wikipedia sea muy similar a herramientas como WordNet. Además puede verse como una especie de tesauro donde se combina etiquetado colaborativo e indexación jerárquica.

La estructura que presentan las categorías es la de un grafo dirigido en que pueden aparecer ciclos, ya que puede suceder que algunas categorías sean simultáneamente supercategorías y subcategorías de una misma categoría. El carácter colaborativo de esta categorización, propicia la aparición de errores como éste, pero también ofrece como aspecto positivo el mantenimiento continuo y la retroalimentación recibida a través de los usuarios. Este tipo de estructuras han sido ampliamente estudiadas en [Mehler, 2009] donde reciben la denominación de *grafos de categorías* o *grafos de ontologías sociales* (*Social Ontology Graph*, *SOG*), donde el término social se refiere a la forma colaborativa en la que se ha construido la estructura. Tomando como referencia la definición que se hace en este trabajo sobre grafos de categorías, definimos el grafo de categorías de Wikipedia de la siguiente manera.

Definición 3. *Definimos el grafo de categorías de Wikipedia G_W como un grafo dirigido $G_W = (V, A, r)$, donde V es un conjunto de vértices, en este caso cada vértice se corresponde con una categoría de Wikipedia, A es una familia de arcos que se corresponderían con las relaciones jerárquicas entre categorías y r representa la categoría principal de Wikipedia que es el nodo raíz en G_W. Para el grafo se definen dos funciones $init : A \to V$ y $ter : A \to V$ asignando a cada arco $a \in A$ un vértice inicial $init(a)$ y un vértice final $ter(a)$. De este modo indicamos que el arco a está dirigido desde $init(a)$ hasta $ter(a)$.*

Adicionalmente es conveniente definir el contenido de una página de Wikipedia.

Definición 4. *Una página de Wikipedia se puede caracterizar como $p = (t, c, L, LC)$, donde t es el título, y c el contenido de la página, $L = \{l_1, l_2, \ldots, l_n\}$ es el conjunto de enlaces que hacen referencia a otras páginas y $LC = \{lc_1, lc_2, \ldots, lc_n\}$ es el conjunto de enlaces que referencian a las categorías en las que se ha clasificado la página.*

En la Figura 5.2 podemos ver una página de Wikipedia con cada uno de sus componentes marcados, título, contenidos, enlaces a páginas y enlaces a categorías.

En una categoría tenemos información referente a las páginas que contiene, así como de sus categorías padre. Cada página de Wikipedia está asociada a una o más categorías, por lo que podemos definir una categoría de Wikipedia como el conjunto de páginas asociadas a ella. Desde el punto de vista de las páginas las categorías representan, en cierta forma, etiquetas que describen los temas sobre los que trata el artículo/página.

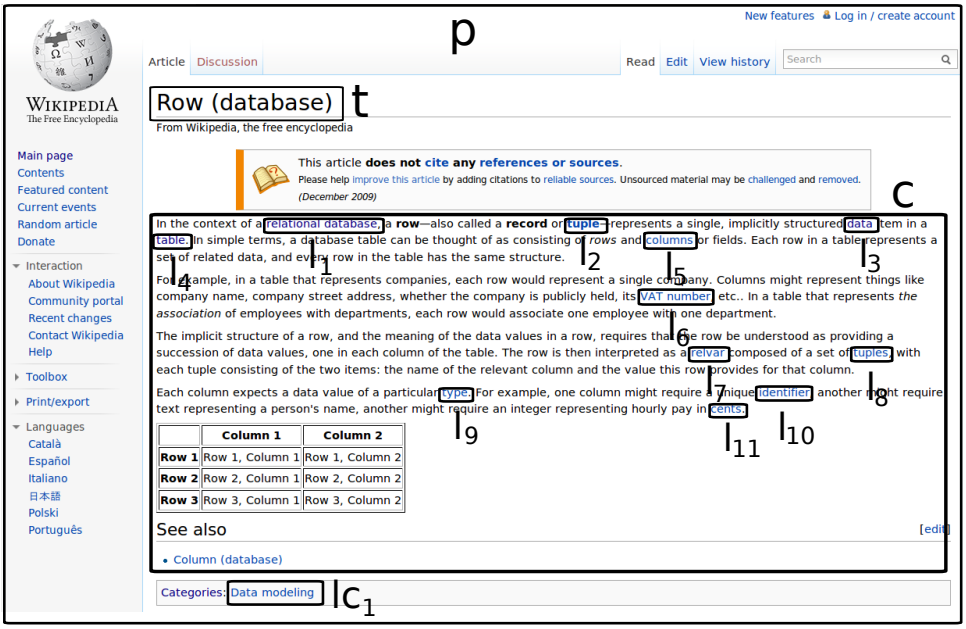


Figura 5.2: Componentes de una página de Wikipedia

Definición 5. *Definimos una categoría de Wikipedia* $C = (P, LC)$ *como un conjunto de páginas o artículos de Wikipedia* $P = \{p_1, p_2, \ldots, p_n\}$ *y un conjunto de enlaces a sus categorías padre* $LC = \{lc_1, lc_2, \ldots, lc_n\}$.

De la definición anterior se desprende que la intersección del conjunto de páginas de una categoría con las de otra no tiene porqué ser vacía, ya que una página puede estar clasificada en dos categorías distintas.

En la Figura 5.3 podemos ver como para una categoría de Wikipedia se tienen un conjunto de enlaces a sus categorías padre, y un conjunto de páginas que están clasificadas como pertenecientes a dicha categoría.

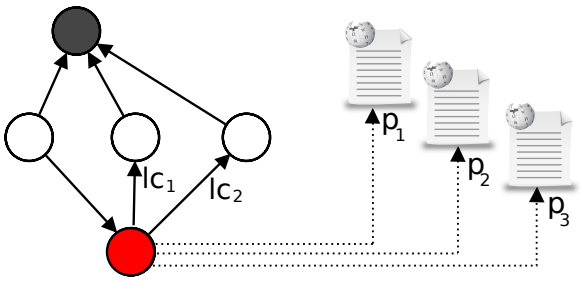


Figura 5.3: Componentes de una categoría de Wikipedia

El grafo de categorías de Wikipedia es de gran tamaño y está altamente conectado. El gran tamaño que presenta es una desventaja cuando se trata de realizar un uso automático. Supongamos que queremos asociar cada uno de los términos de que disponemos en la

Estructura-AP con una categoría, y que a partir de dichas categorías obtuviésemos la jerarquía completa para dichas categorías, el árbol/grafo de categorías obtenido sería muy similar en tamaño al original debido a la alta conectividad existente.

En nuestro caso nos interesa crear una ontología donde se contextualicen los términos que conforman la Estructura-AP, pero dado que esta ontología ha de ser descriptiva del campo textual del que ha sido extraída y que va a ser utilizada y consultada por usuarios humanos, no tiene sentido que sea de gran tamaño. De este modo lo que proponemos es la selección de un sub-árbol dentro del grafo de categorías, que defina de forma adecuada a un término o conjunto de términos de la Estructura-AP. La selección de este sub-árbol se basará en medidas de información, de similitud semántica y de distancia semántica.

Para seleccionar las categorías de Wikipedia que consideramos relevantes necesitamos un mecanismo para establecer una comparación entre ellas. Nuestro trabajo con la estructura de categorías de Wikipedia se va a centrar en la explotación de la información atendiendo a tres enfoques distintos:

- Exploración de grafos: Partiendo de una categoría inicial, establecemos distintos métodos para recorrer el grafo subyacente en orden ascendente, de modo que vamos seleccionados una serie de categorías padre en el camino hacia la raíz. Cada uno de estos métodos serán analizados en secciones siguientes.

- Estructura semántica: Empleando medidas de similitud semántica creadas para WordNet y adaptadas para Wikipedia, podemos obtener valores de similitud entre categorías atendiendo a criterios de tipo topológico sobre la estructura definida. Esto nos permite seleccionar únicamente aquellas categorías relaciondas con las iniciales, con el objetivo de definir una estructura coherente con la información de la que disponemos de partida.

- Recuperación de información: Cada una de las categorías de Wikipedia contiene un conjunto de páginas, éstas a su vez tienen un título y un contenido de la página. Utilizando técnicas empleadas en recuperación de información podemos determinar la similitud de dos categorías atendiendo a su contenido. Para realizar esta comparación podemos actuar a distintos niveles.

    1. A nivel de página: Se toma cada página como una única entidad, de modo que una categoría se ve como un conjunto de páginas. Para poder comparar páginas entre categorías seleccionamos los títulos de las páginas como cadenas completas, de modo que la categoría queda representada por un conjunto de títulos de página.
    2. A nivel de título de página: En este caso se toman todos los títulos de las páginas que hay en una categoría y se extraen cada uno de los términos representándolos en forma de una única bolsa de palabras que define a la categoría. De este modo podemos comparar categorías entre sí.
    3. A nivel de contenido de la página: Este último caso implica mayor procesamiento, puesto que todo el contenido de todas las páginas se expresa como una única

bolsa de palabras asociada a la categoría. Comparar dos categorías se reduce a comparar sus bolsas de términos con alguna de las métricas establecidas para tal efecto.

Una vez introducidas las distintas posibilidades para obtención de información de la estructura que manejamos, más adelante veremos como podemos aplicarlas a la generación de ontologías.

Es importante determinar la idoneidad del uso del grafo de categorías de Wikipedia, para las tareas que vamos a realizar. A este respecto en [Torsten Zesch, 2007] se argumenta la utilidad del uso del grafo de categorías para tareas de procesamiento del lenguaje natural, llegando a la conclusión de que es una herramienta tan fiable como otros diccionarios electrónicos. En ese mismo trabajo se presentan algunas estadísticas con el objetivo de comparar el tamaño y cobertura de Wikipedia con respecto a WordNet. La versión de Wikipedia con la que se realizaron los experimentos correspondientes databa de 15 de Mayo de 2006, más concretamente de la versión alemana de Wikipedia. Únicamente se tuvo en consideración el mayor componente conexo del grafo, el cual contenía un 99.8 % de los nodos existentes. Para esta versión se obtuvieron las siguientes estadísticas, 27865 nodos, el numero medio de arcos conectados a un nodo es de 3.54, un diámetro de 17 arcos (el diámetro es la longitud del mayor de los caminos mínimos) y la media del camino mínimo entre cualquiera dos arcos es de 7.18 arcos. Para establecer un marco de referencia, señalaremos que para esos mismos parámetros, para WordNet se obtienen unos valores de 122005 nodos, un diámetro de 27 arcos y un camino mínimo de una longitud media de 10.56.

5.2. Extensión Semántica de los Conjuntos-AP empleando Wikipedia

Tal y como se ha comentado previamente, para esta tarea, el proceso de generación de la Estructura-AP se realiza de forma idéntica a como se ha realizado en el caso de WordNet. Así pues, nos remitimos a la figura 4.3 que contiene el conjunto de itemsets maximales que definen la Estructura-AP. Partiendo de esta Estructura-AP, más concretamente del retículo definido por ella, procedemos a obtener una ontología básica de inclusión, tal y como se mostró de forma resumida en la Figura 4.8.

El primer método y más sencillo consiste en la búsqueda de los términos que conforman cada uno de los conceptos en la jerarquía básica de inclusión. Se recuperan las páginas asociadas a los términos y a partir de ellas se obtienen las categorías asociadas a dichas páginas. Una vez tenemos un conjunto de categorías base de partida, recorremos el grafo de categorías de Wikipedia conectando las categorías base obtenidas inicialmente.

En la Figura 5.4 podemos ver de forma gráfica la abstracción de la correspondencia establecida entre cada uno de los Conjuntos-AP obtenidos para la Estructura-AP y las categorías de Wikipedia. El proceso completo consistiría en generar la ontología básica de inclusión con las reglas sintácticas presentadas en la Sección 4.4.5 y luego asociar cada una de las clases obtenidas a una categoría de Wikipedia. Este paso intermedio se realiza para

dotar de cierto orden a los items contenidos dentro de cada Conjunto-AP, pero a alto nivel lo único que estamos haciendo, tal y como muestra la figura, es asociar cada Conjunto-AP de la Estructura-AP a una categoría, siempre que esto sea posible.

Para establecer esta asociación buscamos páginas cuyo título se corresponda con el nombre de las clases de la ontología básica de inclusión, y por tanto con los Conjuntos-AP correspondientes. Una vez establecida la correspondencia, entendemos que la página define el concepto representado por el Conjunto-AP. Una vez obtenidas todas las páginas, obtenemos la categoría en que están clasificadas y comenzamos a recorrer el grafo correspondiente a las categorías en dirección a la raíz. En el caso de que no se encontrase ninguna página cuyo título se corresponda con el buscado, continuamos con el procesamiento, y en la ontología final sólo se incluirá la información disponible en la ontología básica de inclusión. Dado que una página puede estar clasificada en varías categorías, debemos seleccionar aquella más apropiada, en nuestro caso ésta será la que tenga un nombre más parecido al de la página correspondiente.

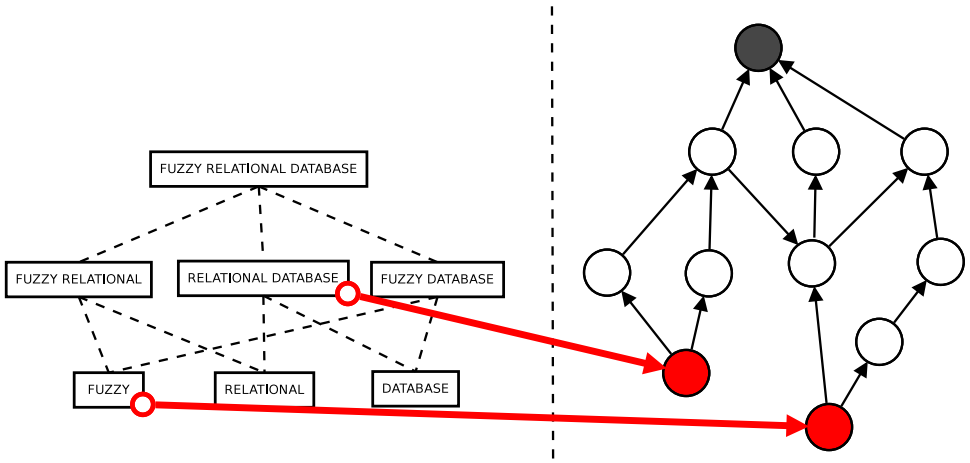


Figura 5.4: Correspondencia de la Estructura-AP con grafo de Wikipedia

Debido a la alta conectividad del grafo de categorías de Wikipedia, será necesario emplear distintos criterios para la selección de los caminos apropiados entre pares de categorías base. Los métodos empleados se describen en la siguiente sección.

Para la realización de las pruebas y la obtención de resultados hemos trabajado sobre un volcado de Wikipedia en inglés, correspondiente a la fecha de 17 de Agosto de 2010, almacenado en una base de datos MySQL.

5.2.1. Métodos Basados en Exploración de Grafos

En este apartado vamos a presentar distintas variantes a la hora de recorrer el un grafo desde un punto determinado inicialmente hasta la raíz. Dado que cada uno de los términos seleccionados en los Estructura-AP representa una categoría, y por tanto, un nodo en el grafo, buscaremos caminos coherentes hacia la raíz del grafo, seleccionado aquellos nodos que aporten información relativa a las categorías inicialmente seleccionadas.

Método de Exploración Básico

Este método no utiliza ningún tipo de procesamiento adicional, parte de cada una de las categorías y explora el grafo de forma ascendente. Para evitar la aparición de ciclos se mantiene una lista de nodos visitados. Si un nodo ha sido visitado anteriormente no se vuelve a procesar.

Utilizando este método obtenemos una estructura jerárquica que contiene todas las categorías correspondientes a los términos que aparecían en la Estructura-AP. Si bien la ontología obtenida es correcta, debido a su enorme tamaño no resulta práctica.

En la Figura 5.5 podemos ver un ejemplo de cómo se explora el grafo con este método. Los nodos coloreados son los nodos de partida, y aquellos cuyo borde es más grueso son los nodos visitados en el camino hacia la raíz, que está coloreada en negro. Como podemos ver en este ejemplo, ante una situación de alta conectividad, el número de nodos no visitados al final del proceso es mínimo, seleccionando la práctica totalidad de los nodos existentes.

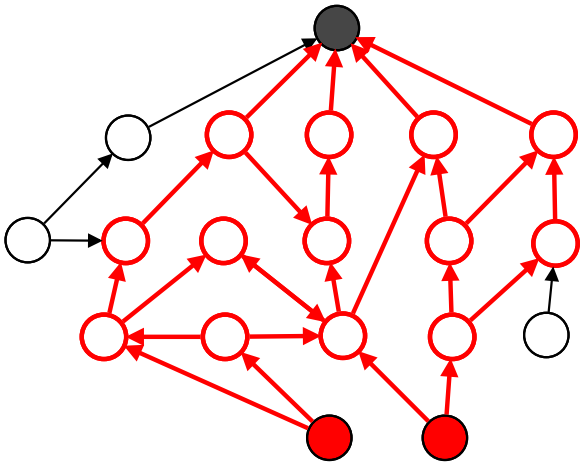


Figura 5.5: Método Básico de exploración del grafo

La descripción formal del proceso realizado puede encontrarse en el Algoritmo 5.

Partiendo del mismo conjunto de datos planteado en secciones anteriores obtenemos la Estructura-AP que ya vimos en la Figura 4.4. Para simplificar un poco las distintas pruebas que vamos a ir realizando seleccionaremos dos elementos del retículo. Tomaremos el elemento del primer nivel *SQL* y el elemento del segundo nivel *Association Rule* con el objetivo de poder explicar de la forma más clara posible las diferentes aproximaciones y posteriormente poder extender estos casos básicos a otros más complejos y generales.

Según el método básico descrito obtendríamos una ontología con las características que se muestran en la Tabla 5.1.

Como se puede observar el alto número de clases seleccionadas en el camino a la raíz es excesivo. Esto se debe a que las zonas altas del grafo de categorías de Wikipedia contienen conceptos muy generales y altamente conectados entre sí. De este modo consideramos que debido a la gran conectividad existente, es un dato muy interesante conocer el camino mínimo y el máximo desde cada una de las categorías correspondientes a los términos que

Algoritmo 5 Algoritmo de Exploración Básico

Input: El grafo $G_W = (V, A, r)$ y un conjunto de nodos iniciales $Vc = \{v_1, \ldots v_n\}$
Output: El conjunto de nodos Vs y el conjunto de arcos As

$Vs_0 \leftarrow \emptyset, As_0 \leftarrow \emptyset$
$Vs_1 \leftarrow v_l, As_1 \leftarrow \emptyset$
$k \leftarrow 2$
for all $\{v_i \in Vs_{k-1} \wedge v_i \notin \bigcup_{j=0}^{k-2} Vs_j\}$ **do**
  if $\{a \in A, v \in V : (init(a) = v_i) \wedge (ter(a) = v)\}$ **then**
    $As_k \leftarrow a, Vs_k \leftarrow v$
  end if
  $Vs_k \leftarrow [Vs_k \backslash Vs_{k-1}]$
  $k++$
end for
$As \leftarrow \bigcup_{i=1}^{k-1} As_i, Vs \leftarrow \bigcup_{i=1}^{k-1} Vs_i$
return As, Vs

Tabla 5.1: Número de clases obtenidas para la ontología generada siguiendo el método básico

	SQL	*Association Rule*
Categoría Base	SQL	Data Mining
Número de Nodos/Categorías	2786	2784
Número medio de Nodos Padre	8	8
Número máximo de Nodo Padre	14	14
Número medio de Nodos Hermanos	15	15
Número máximo de Nodos Hermanos	42	42
Profundidad Máxima	56	53
Profundidad Mínima	5	3

estamos empleando. Debemos recordar que los nodos del grafo sólo se visitan y se expanden una única vez, para evitar la aparición de bucles.

Dado que este método no es de utilidad para ser empleado en un recurso como Wikipedia, proponemos una alternativa, que desarrollamos a continuación.

Método Básico Resumido

Este método parte como el método básico de un conjunto de categorías seleccionadas a través de los términos. En este caso en lugar de obtener todo el árbol que contiene los términos, lo que se va a hacer es buscar la categoría padre común más específica que contiene a las categorías actuales. En [Wu and Palmer, 1994] se introduce este concepto que se denomina mínimo superconcepto común, o mínimo subsumidor común (*least common subsumer*, *lcs*). El conjunto de categorías originales para las cuales estamos buscando padres comunes las llamaremos *categorías base*. Las categorías seleccionadas como parte del proceso de detección de categorías comunes las denominaremos *categorías padre comunes*.

De este modo, eliminamos todas las categorías intermedias y nos quedamos únicamente con las comunes a una o más de las categorías iniciales de las que partíamos. Cada vez que encontremos una categoría padre que contiene a varias categorías, volvemos a repetir el proceso, pero en este caso empleando la categoría padre. Para evitar los ciclos, igual que en el caso anterior iremos registrando una lista de nodos visitados.

En la Figura 5.6 podemos ver para nuestro grafo de ejemplo cómo se aplica el algoritmo. Los nodos coloreados indican las posiciones iniciales. En líneas gruesas aparecen los caminos sobre el grafo hasta el *lcs* y en línea discontinua los nuevos arcos creados que no pertenecen al grafo original.

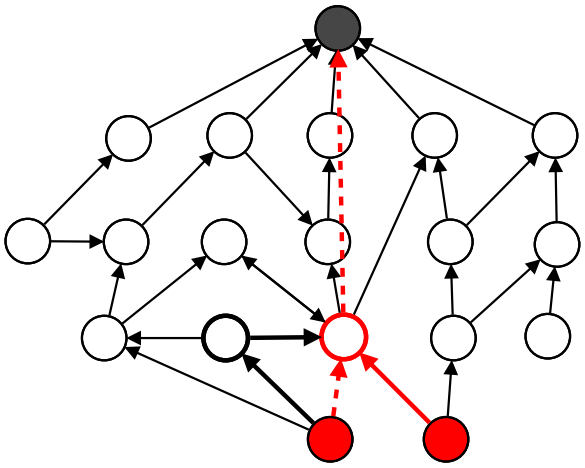


Figura 5.6: Método de exploración del grafo Básico Resumido

La descripción formal del proceso se indica mediante el Algoritmo 6, teniendo en cuenta que el nodo devuelto por *lcs* nunca va a ser la raíz (r). Si el *lcs* es la raíz, se entiende que no hay *lcs* entre los dos nodos.

Algoritmo 6 Algoritmo de Exploración Básico Resumido

Input: El grafo $G_W = (V, A, r)$ y un conjunto de nodos iniciales $Vc = \{v_1, \ldots v_n\}$
Output: El conjunto de nodos Vs y el conjunto de arcos As

$Vs_1 \leftarrow Vc, As_1 \leftarrow \emptyset$
$k \leftarrow 1$
while $Vs_k \neq \emptyset$ **do**
  for all $\{v_i \in Vs_k\}$ **do**
    for all $\{v_j \in Vs_k : j > i\}$ **do**
      $l \leftarrow lcs(v_i, v_j)$
      if $l \neq \emptyset$ **then**
        $Vs_{k+1} \leftarrow l$
        $As_{k+1} \leftarrow b : init(b) = v_i \wedge ter(b) = l$
        $As_{k+1} \leftarrow b : init(b) = v_j \wedge ter(b) = l$
      end if
    end for
  end for
  $Vp = \{b \in As_{k+1}, v \in Vs_k, w \in Vs_{k+1} : (init(b) = v) \wedge (ter(b) = w)\}$
  $Vs_k \leftarrow [Vs_k \cup Vs_{k+1}] \backslash Vp$
  $k++$
end while
for all $\{v_i \in Vs_{k-1}\}$ **do**
  $As_{k-1} \leftarrow b : init(b) = v_i \wedge ter(b) = r$
end for
$As \leftarrow \bigcup_{i=1}^{k-1} As_i, Vs \leftarrow \bigcup_{i=1}^{k-1} Vs_i$
return As, Vs

Con este método evitamos que la ontología resultante adquiera grandes dimensiones, pero por otra parte, perdemos mucha de la terminología que aportaban las categorías intermedias. Otro inconveniente es que uno de los términos forme parte de una categoría ancestro de las categorías de otros términos, en este caso el *lcs* será la categoría ancestro, y por tanto no tendremos nuevas categorías en el resultado final. También es importante resaltar que en ocasiones puede que las categorías comunes sean generales y poco relevantes por lo que no aportan información alguna. Del mismo modo, al realizar el procesamiento de las categorías hasta los niveles superiores del grafo de categorías, podemos incluir terminología genérica y algo alejada del dominio al que se circunscriben los términos que estamos empleando.

En la Figura 5.7 podemos ver la ontología obtenida aplicando este método sobre los términos *SQL* y *Association Rule*. Podemos ver como se han encontrado clases comunes que permiten dotar de mayor semántica a la ontología, pero no terminan de aportar suficiente información como para considerar que con este método basta para generar ontologías relevantes para los términos.

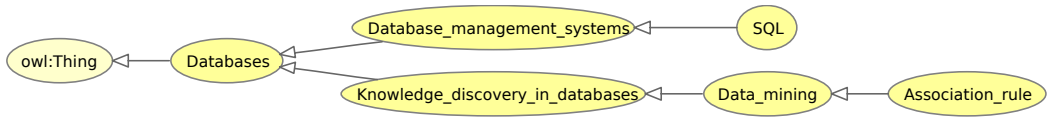


Figura 5.7: Ancestro común

Métodos con Longitud de Camino

Dado que el principal inconveniente con el que nos encontramos es el gran tamaño de la estructura de partida, una forma rápida y sencilla de eliminar este problema es determinar la longitud máxima del camino a explorar. Ofrecemos como parámetro de entrada al algoritmo un número entero que indica la longitud máxima n que pueden tener los caminos que llevan a una clase padre. De este modo se poda por la parte superior, eliminando categorías mas generales.

Esta aproximación tiene diversas ventajas, pero los inconveniente son también bastante importantes.

Como ventaja nos encontramos con que el método elimina una gran cantidad de categorías, haciendo la ontología más manejable. Además el uso del parámetro n nos garantiza un tamaño acotado de profundidad en la ontología final.

Este método no tiene en cuenta más que la topología de la red, es de suponer que nodos cercanos estén relacionados, pero puede que esta relación sea muy general, o incluso tangencial. Si queremos seleccionar de forma adecuada las categorías debemos tener en cuenta medidas de carácter semántico. La distancia entre los nodos de un grafo puede ser un buen complemento a otro tipo de medidas, pero como tal no garantiza la preservación del contexto de los términos iniciales.

Uno de los inconvenientes más importantes podemos encontrarlo en que existan una o varias categorías comunes para ciertas categorías base, pero que estas se encuentren a

una longitud de camino superior a la permitida como parámetro. Así podemos tener dos variantes del método, un método básico con longitud de camino, y otro método básico resumido con longitud de camino.

Método Básico con Longitud de Camino

Este método parte de las categorías base e incluye todas las categorías ancestro que se encuentren a una distancia menor o igual a la indicada como parámetro de entrada n. Toda categoría que se encuentre a una distancia superior al parámetro n va a ser ignorada.

En la Figura 5.8 se muestra el resultado para el grafo de ejemplo, aplicando el método para una longitud de camino $n = 2$. Los nodos coloreados representan los puntos de partida de la exploración y los arcos resaltados con un trazo más grueso son aquellos que han sido seleccionados. Los arcos mostrados con línea discontinua, identifican nuevos arcos creados para conectar con la raíz. Como se puede observar se han incluido algunos nodos que se encuentra a una distancia mayor, pero esto se debe a que existe algún camino alternativo a la distancia indicada.

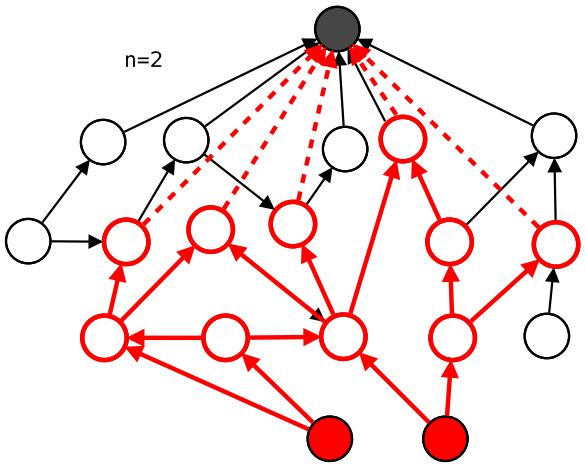


Figura 5.8: Método de exploración del grafo Básico con longitud de Camino

La descripción formalizada del proceso se muestra en el Algoritmo 7.

A continuación vamos a ver cómo el cambio del parámetro n afecta a los resultados obtenidos.

En la Figura 5.9 podemos observar la ontología generada para una longitud de camino $n = 6$, para el término *SQL*. Como se puede observar, con una longitud medianamente alta, ya nos situamos en las zonas superiores del grafo de categorías de Wikipedia. Podemos observar que cuanto mas ascendemos por el grafo, más se va ramificando este, y mas conectado se encuentra. Dado que las categorías que aparecen con esta longitud ya son bastante generales, nos planteamos reducir el valor de n.

Antes de continuar, repetimos el proceso para el término *Association Rule*, en esta ocasión y para un valor de $n = 6$ obtenemos un total de 216 clases. Esto se debe principalmente a que, de nuevo, se alcanzan las categorías superiores donde aumenta la conexión y

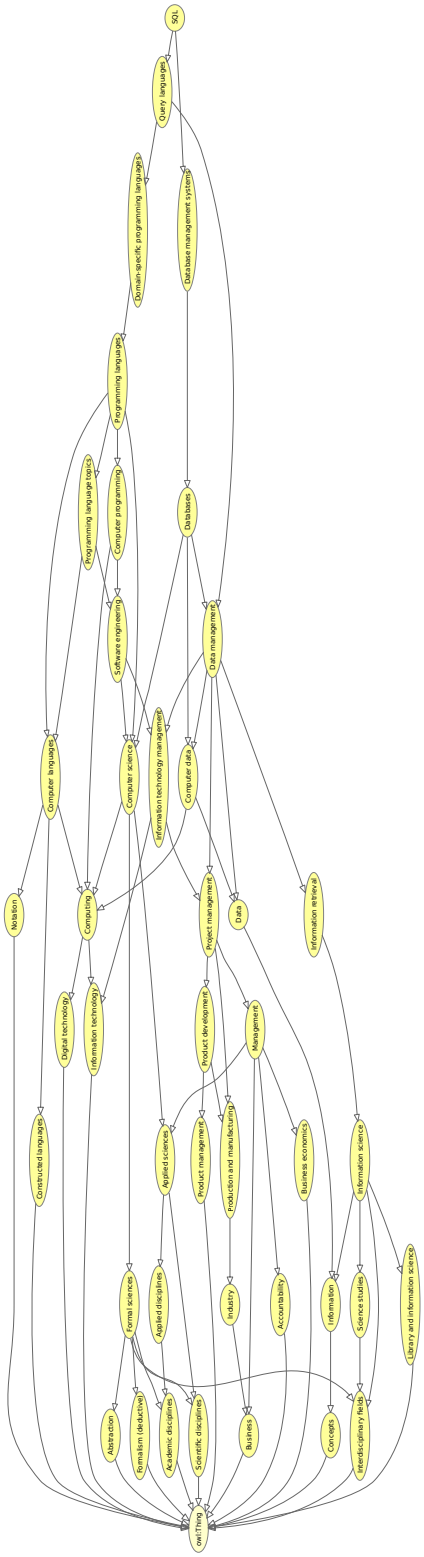


Figura 5.9: Resultado para el término SQL con el parámetro n=6

Algoritmo 7 Algoritmo de Exploración Básico con longitud de Camino

Input: El grafo $G_W = (V, A, r)$, los nodos iniciales $Vc = \{v_1, \dots v_n\}$ y una longitud n
Output: El conjunto de nodos Vs y el conjunto de arcos As

$Vs_0 \leftarrow \emptyset, As_0 \leftarrow \emptyset$
$Vs_1 \leftarrow Vc, As_1 \leftarrow \emptyset$
for $k \leftarrow 2$ **to** $n+1$ **do**
  for all $\{v_i \in Vs_{k-1} \wedge v_i \notin \bigcup_{j=0}^{k-2} Vs_j\}$ **do**
    if $\{a \in A, v \in V : init(a) = v_i \wedge ter(a) = v\}$ **then**
      $As_k \leftarrow a, Vs_k \leftarrow v$
    end if
    $Vs_k \leftarrow [Vs_k \backslash Vs_{k-1}]$
  end for
end for
$As \leftarrow \bigcup_{i=1}^{n+1} As_i, Vs \leftarrow \bigcup_{i=1}^{n+1} Vs_i$
return As, Vs

la abstracción de los conceptos. Además, la categoría asociada al término *Association Rule* es *Data Mining*, la minería de datos es una materia multidisciplinar que entronca con otras ramas del conocimiento, por lo que a la amplia cobertura en anchura debemos añadir el aumento de la conectividad al acercarnos a la raíz, esto provoca la aparición de un número muy elevado de clases en la ontología final, lo que implica la generación de diagramas que no son fácilmente representables. Para poder mostrar un gráfico adecuado modificaremos el valor n hasta obtener un número adecuado de nodos, que puedan ser representados, en este caso seleccionamos el valor $n = 3$, tal y como se muestra en la Figura 5.10.

Vemos cómo para ambos términos los resultados obtenidos difieren, hasta tal punto que con tres niveles menos de exploración para el término *Association Rule* se aproxima bastante el número de nodos representados a los obtenidos para el término *SQL* con una longitud mayor. A lo largo de una serie de experimentos vamos a comprobar que este no es un hecho aislado, y que por lo tanto no es posible determinar un valor adecuado para n de forma experimental, ya que este dependerá de la forma en que el término esté conectado. En este caso *SQL* es un término claramente categorizado que pertenece a una disciplina concreta, y por lo tanto tenemos una estructura en árbol con pocos nodos hermanos, por el contrario *Association Rule* al estar en la categoría *Data Mining* cuyo carácter es multidisciplinar, se extiende a lo ancho y provoca que cambios mínimos en el parámetro n provoquen la aparición de multitud de nuevos nodos.

Para poder observar mejor el comportamiento del método según se incrementa el parámetro n, en la Figura 5.11 se muestra la cantidad de clases que presenta la ontología final atendiendo al valor n de longitud de camino seleccionado. Como podemos ver el comportamiento de ambas gráficas es prácticamente igual, la única diferencia es que la correspondiente a *SQL* se encuentra desplazada a la derecha tres unidades. Este compor-

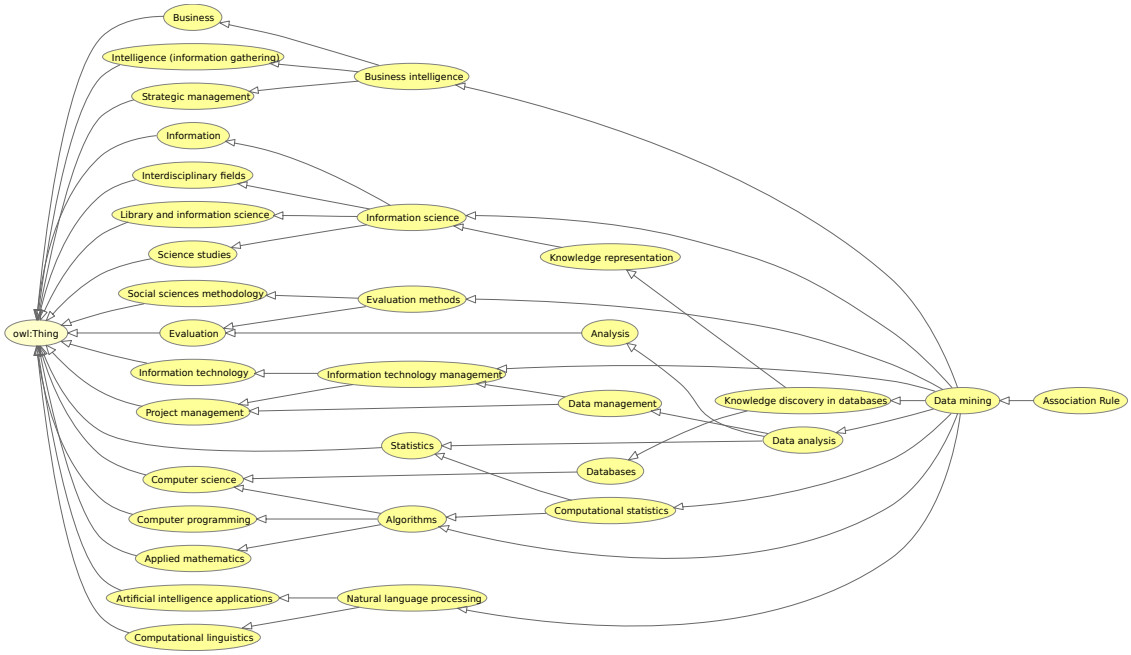


Figura 5.10: Resultado para el término Association Rule con el parámetro n=3

tamiento puede explicarse en base al método empleado para explorar el grafo y la forma de expandir los nodos, dado que este método es determinista, un comportamiento tan similar nos lleva a pensar que en algún punto se ha llegado a una categoría común, y que la expansión a partir de ese momento es idéntica. El desplazamiento vendría dado por la diferencia de profundidad a la que se encuentran las categorías originales. A partir de la primera categoría en común que tienen ambos términos el comportamiento viene a ser el mismo en ambos casos. Esto nos hace plantearnos la importancia de detectar las categorías comunes a las categorías básicas procesadas.

Por otra parte, si analizamos con detenimiento el crecimiento de la gráfica vemos que existe un punto a partir del cual el crecimiento es muy pronunciado, esto se debe principalmente a que se entra en la zona superior donde la alta conectividad provoca la aparición de un alto número de nuevas categorías. Analizaremos este fenómeno más adelante con la ayuda de datos adicionales. Pasado el punto de inflexión de crecimiento rápido, la gráfica sigue un crecimiento casi lineal con ligeras variaciones hasta llegar a una cota superior. Esta cota viene dada por el número total de nodos que se seleccionarían en el caso de que no se estableciese ningún criterio, lo que corresponde a nuestro primer caso con el método básico. Vemos como efectivamente los valores obtenidos entonces, se corresponden con la cota superior obtenida experimentalmente, por lo que podemos deducir que el camino mas largo dentro del grafo se corresponde con el valor de la cota, en este caso 56 para el término *SQL* y 53 para el término *Association Rule*.

Como vemos en ambos casos, la cota superior del número de clases generadas es muy alto, resultados con tantas clases no aportan información alguna, de modo que nos cen-

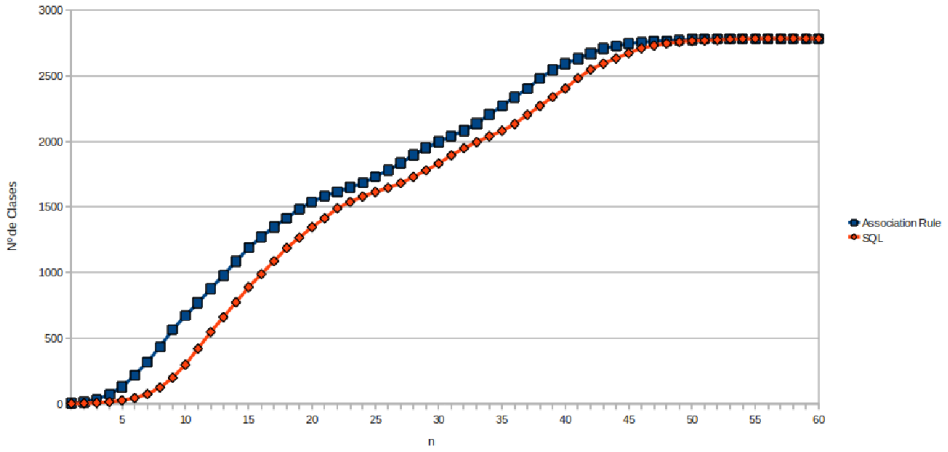


Figura 5.11: Número de Clases para los términos *SQL* y *Association Rule*, con distintos valores de n

traremos en tamaños de camino relativamente pequeños. En la Figura 5.12 se muestran valores para longitudes de camino de hasta $n = 20$. En esta gráfica vemos como para tener un tamaño de ontología final manejable que incluya menos de 100 clases, el valor de n debe estar cercano a 4. En cualquier caso este valor dependerá de la profundidad de la categoría inicial, y el número de arcos que la conecten con una categoría general que provoque que entre en la zona superior del grafo.

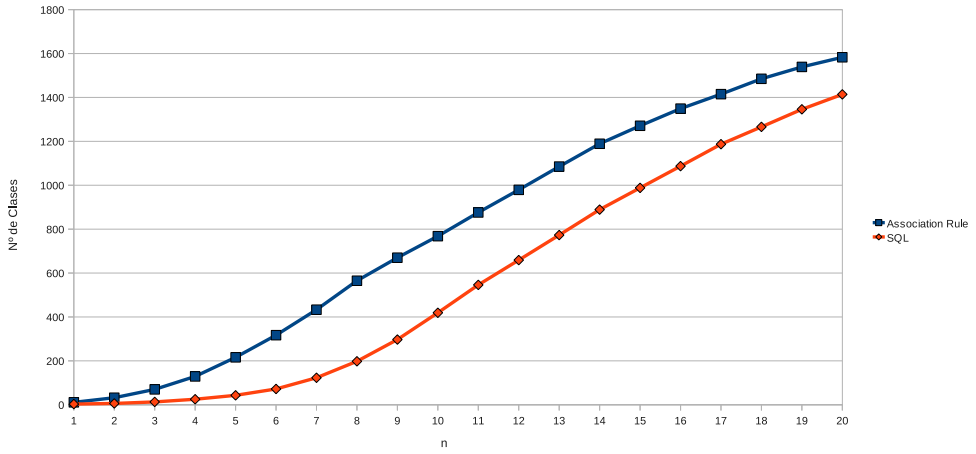


Figura 5.12: Vista detallada para caminos de longitud 20

Así pues, vemos que el principal problema que encontramos para utilizar este método, es adentrarnos en la parte superior del grafo y entrar en la zona que incluye categorías abstractas. Como hemos visto en los casos anteriores, esto dependerá de cada término y no es algo que se pueda determinar de forma sencilla. Para intentar analizar este fenómeno con mayor detenimiento, en la Figuras 5.13 y 5.14 mostramos algunos valores estadísticos

asociados a las ontologías obtenidas para diferentes longitudes de camino n. Se muestra el número medio y máximo de clases padre que tiene una clase cualquiera, y el número medio y máximo de clases hermanas. A través de estos datos podemos averiguar el momento en que se comienza la exploración de la zona superior del grafo, puesto que el número de clases hermanas y el número de padres aumentará de forma notable.

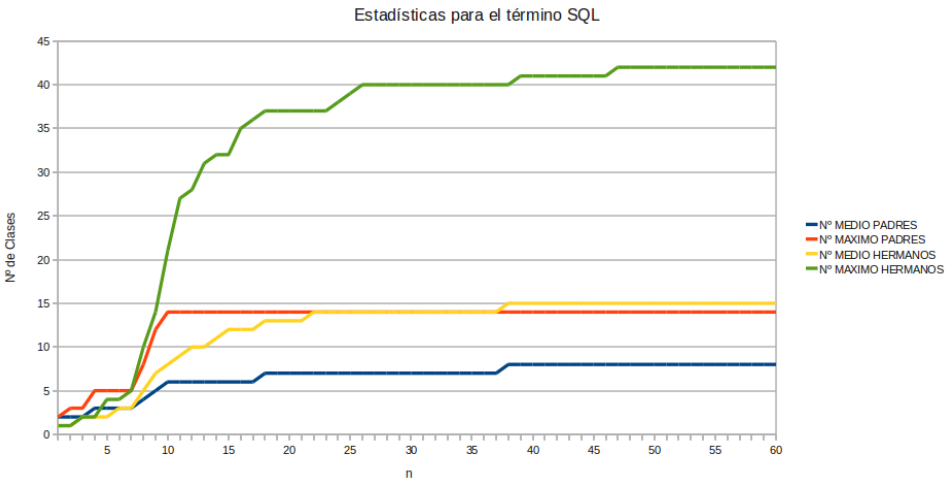


Figura 5.13: Estadísticas para el término *SQL*

En la Figura 5.13 Para el término *SQL* comprobamos que a partir de un camino $n = 7$ los valores promedio y máximos se disparan lo que indicaría que por cada nueva categoría a explorar se incluirían una gran cantidad de categorías padre. Esto se ve reflejado también en la Figura 5.12 donde se aprecia un notable incremento del número de clases en la ontología final para un valor $n = 7$.

Para el término *Association Rule*, las estadísticas mostradas en la Figura 5.14 son muy parecidas a las obtenidas para *SQL*, esta ocasión, el punto de máximo crecimiento se encuentra para $n = 4$.

Hemos podido comprobar como es necesario conocer muy bien el contexto para poder fijar un valor adecuado de n. Un dato que puede ser de gran ayuda es la profundidad a la que se encuentra la categoría, de este modo se puede ajustar el valor, evitando alcanzar las categorías superiores que son más generales. Sin embargo, hemos visto que la aparición de categorías comunes a los términos determina de forma clara el resultado final, por lo que nos centraremos en esta idea en la siguiente sección.

Método Básico Resumido con Longitud de Camino

Este método se basa en el método básico resumido, y lo que hace es expandir el conjunto de categorías intermedias obtenidas entre la clase base y la categoría padre común con otra categoría. Partiendo del conjunto de categorías obtenidas con el método básico resumido, se incluyen nuevas categorías intermedias en base a un parámetro n que determina el número máximo de categorías a mostrar en un camino entre una categoría base y la categoría padre

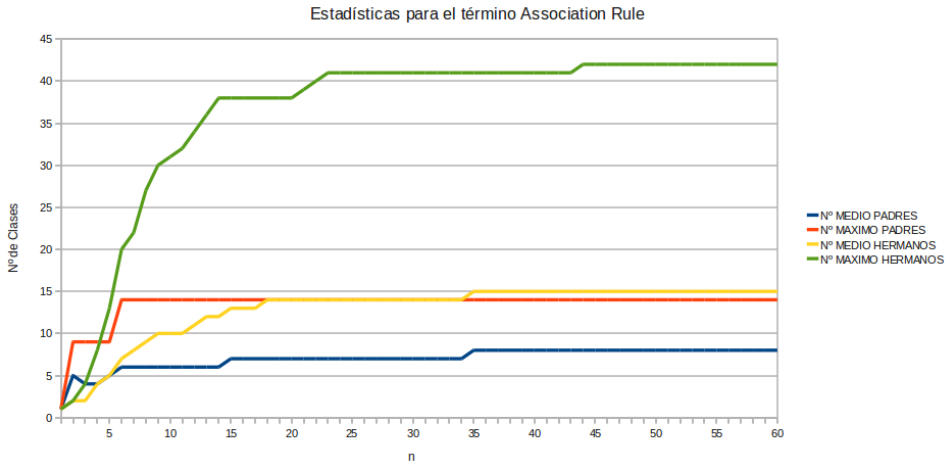


Figura 5.14: Estadísticas para el término *Association Rule*

compartida.

El parámetro n va a determinar cuantas categorías vamos a incluir en el camino entre dos categorías ya determinadas mediante el método básico resumido. La forma en que seleccionamos esas n categorías determinará la organización de la información disponible en la ontología.

Atendiendo a la forma de selección de las categorías intermedias tendremos las siguientes variantes:

1. Seleccionar las n supercategorías inmediatamente superiores: De esta forma se consigue especificar más el contexto cercano a las categorías base, y las categorías padre compartidas sólo se emplean para contextualizar las relaciones entre categorías base.

2. Seleccionar las n categorías inmediatamente inferiores a la común: De esta forma se consigue especificar más el contexto cercano a la categoría común, dejando sin especificar el contexto de las categorías base.

3. Seleccionar las n categorías entre las mas cercanas a la categoría base y las más cercanas a la categoría común: Con esta variante lo que pretendemos es contextualizar en la medida de lo posible las categorías originales ignorando en cierta forma la descripción del camino existente entre ellas. Nos centramos en la vecindad próxima de las categorías, para ofrecer más detalles. En caso de que n sea impar, se seleccionan más categorías de las cercanas a la categoría de origen.

4. Seleccionar n categorías del camino total entre dos categorías seleccionadas: Lo que se pretende con esta variante, es caracterizar de forma adecuada el camino existente entre dos categorías, mostrando una especie de descripción de la relación existente entre ambas categorías.

También es posible indicar que se expandan todos los caminos, independientemente de si llevan a una categoría básica, o no. Así también se expandirían los caminos entre categorías padre relacionadas.

En la Figura 5.15 podemos ver un grafo de ejemplo, y los nodos seleccionados sobre él al emplear los métodos descritos. Los nodos coloreados representan los nodos iniciales, el *lcs* está resaltado con un trazo más grueso. Las líneas gruesas representan arcos que ya existían y que han sido seleccionados, y las líneas gruesas discontinuas representan nuevos arcos creados.

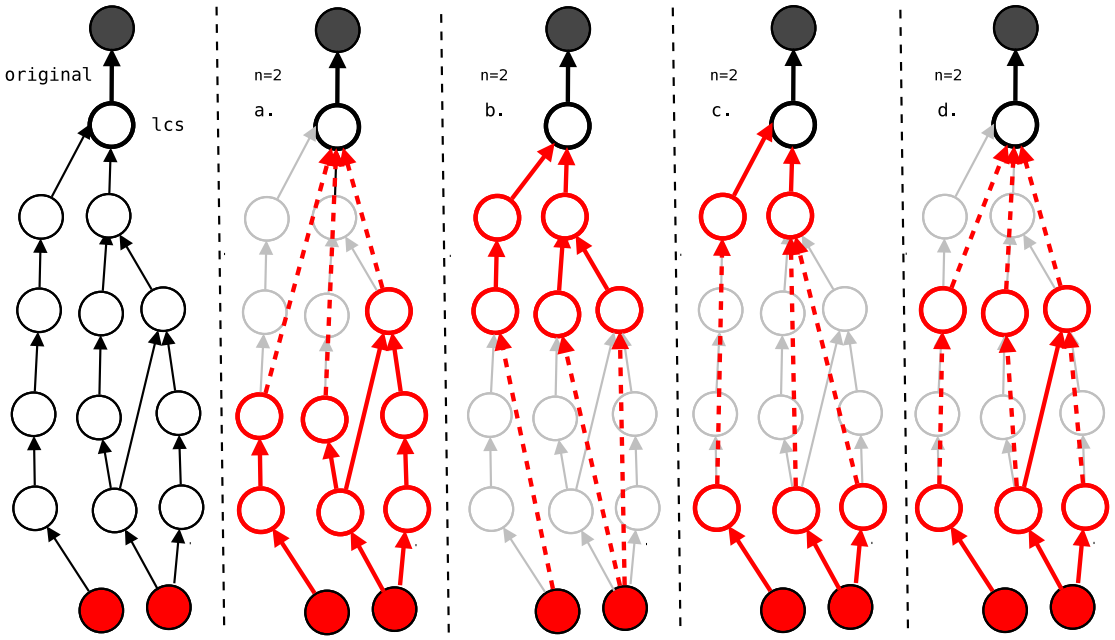


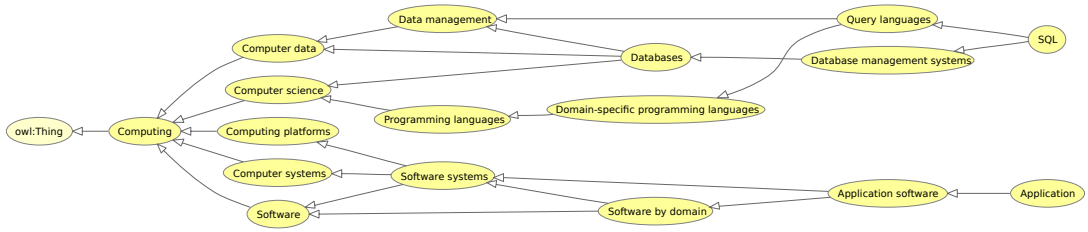
Figura 5.15: Resultados para las distintas variantes del método Básico Resumido con Longitud de Camino

Conviene resaltar que las distintas variantes se aplican sobre cada uno de los posibles caminos entre los nodos iniciales y el *lcs*, ya que cada uno de estos caminos contribuye con el nodo.

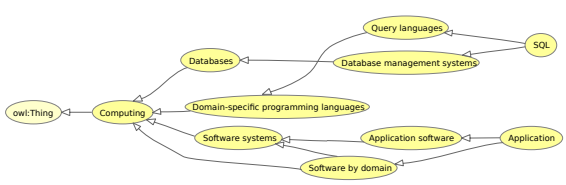
En la Figura 5.16 podemos ver la ontología obtenida para los términos *SQL* y *Application* utilizando el método básico resumido, así como las ontologías resultantes para un valór de parámetro $n = 2$ y los métodos mencionados en el párrafo anterior.

Como se puede observar aunque todos los métodos producen salidas correctas, parece que el primer método ofrece una visión más coherente y contextualizada de los términos utilizados. De nuevo podemos comprobar, como el aumento de la generalización no aporta información relevante. Así pues parece más interesante explorar las categorías cercanas a las seleccionadas inicialmente, en lugar de explorar categorías genéricas.

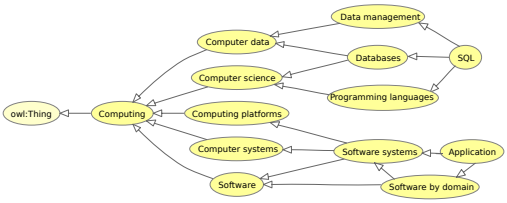
Por otra parte, también existe la posibilidad de especificar el camino desde el *lcs* hasta la raíz, por ejemplo con longitud de camino, pero dado todos los problemas que conlleva internarse en las zonas superiores del grafo, parece más idóneo buscar el camino mínimo



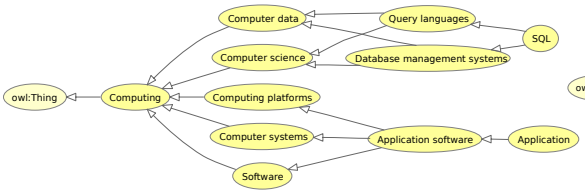
LCS



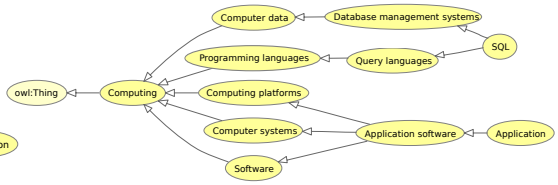
LCS n=2 Método a



LCS n=2 Método b



LCS n=2 Método c



LCS n=2 Método d

Figura 5.16: Resultados para los términos *SQL* y *Application*

hasta la raíz para evitar que la exploración se expanda horizontalmente.

Método Básico de Camino Mínimo

En este método lo que vamos a hacer es buscar el camino mínimo desde cada uno de los nodos a la raíz, sin tener en cuenta los otros términos. Este método tiene el inconveniente de que obviamos categorías padre comunes, que permiten realizar una taxonomía que aporta bastante información.

En la Figura 5.17 podemos ver en el grafo de ejemplo cómo se seleccionarían los caminos mínimos. Los nodos iniciales se muestran coloreados, y cada uno de los nodos y arcos seleccionados se muestran con un trazo más grueso. Vemos como en el nodo inicial de la izquierda se tienen dos caminos mínimos de igual longitud, y por tanto seleccionamos ambos.

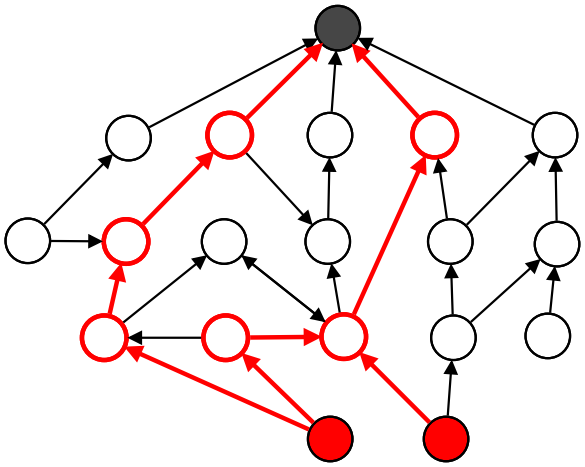


Figura 5.17: Método de exploración del grafo con Camino Mínimo

El Algoritmo 8 presenta una rutina para obtener los caminos mínimos a la raíz a partir de un conjunto de nodos iniciales. El algoritmo hace uso de un procedimiento recursivo para la exploración del grafo. La definición de este procedimiento la podemos encontrar en el Algoritmo 9. La salida del procedimiento será el camino mínimo de entre todos los caminos explorados hasta el momento. La salida del algoritmo general será un conjunto de vértices y arcos que conforman un subgrafo del original que incluye los caminos mínimos hasta la raíz desde cada uno de los vértices iniciales.

Veamos como se comporta el método con los ejemplos reales que hemos ido viendo hasta ahora.

En la Figura 5.18 podemos ver el resultado obtenido para el término *SQL*. Como en casos anteriores, dado que el término se corresponde con la categoría no duplicamos su aparición. Podemos ver que existen dos caminos mínimos que llevan hasta la raíz, que tienen la misma longitud de cinco arcos. En el caso de que exista más de un camino con longitud mínima, tomamos todos los caminos. Ahora mismo no tiene sentido elegir entre

Algoritmo 8 Procedimiento recursivo para buscar el camino mínimo

Input: El grafo $G_W = (V, A, r)$, los nodos iniciales $Vc = \{v_1, \ldots v_n\}$
Output: El conjunto de nodos Vm y el conjunto de arcos Am conteniendo el camino mínimo

```
Vs, As ← ∅
Vp, Ap ← ∅
Vm, Am ← ∅
for all {v_i ∈ Vc} do
    exploraCamino(r, v_i, Vp, Ap, Vs, As)
    Vm ← Vs
    Am ← As
end for
return Vm, Am
```

ninguno de los posibles caminos mínimos, puesto que el criterio que estamos siguiendo es meramente taxonómico, y atendiendo a este criterio ambos caminos son igual de válidos.

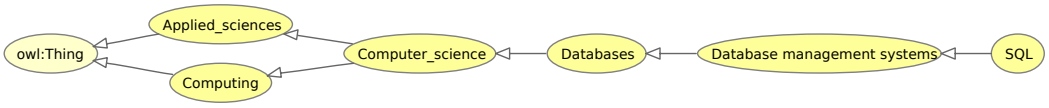


Figura 5.18: Caminos mínimos del término SQL

En la Figura 5.19 podemos ver el resultado obtenido para el término *Association Rule*. Dado que el término no existe como categoría, lo representamos como una clase hija de la clase definida por la categoría a la que pertenece, en este caso *Data Mining*. En esta ocasión vemos que la longitud mínima del nodo a la raíz es de tres arcos, no contabilizaremos el arco entre *Data Mining* y *Association Rule*, puesto que lo hemos definido a posteriori, el conteo siempre se realiza desde la categoría asociada al término hasta la raíz. En este caso observamos que el camino encontrado es disjunto con respecto al encontrado para el término *SQL*. Esto ocurre por tener únicamente en cuenta los caminos mínimos de cada término individualmente en lugar de como un conjunto.

Figura 5.19: Camino mínimo del término Association Rule

Como hemos visto anteriormente en la Figura 5.7 existe un antecedente común para los términos procesados, por lo que parece lógico que esta información, que no solo es relevante, sino que nos permite obtener un grafo conexo, aparezca en la ontología final. Así pues, como condición adicional, pedimos que una vez obtenidos los caminos mínimos para los términos, se obtengan los ancestros comunes, en caso de haberlos entre los distintos nodos del grafo. Es decir, vamos a tratar de conectar los distintos caminos a través de ancestros comunes que

Algoritmo 9 Procedimiento recursivo para buscar el camino mínimo

Procedimiento $exploraCamino(r, c, Vc, Ac, Vs, As)$

Input: El nodo raíz r, el nodo actual c, el conjunto de nodos y arcos actuales Vc y Ac, y el conjunto de nodos y arcos que forman el camino mínimo actual Vs y As

```
Vc ← c
if {c = r} then
  if {|Vs| ≠ 0} then
    if {|Vc| < |Vs|} then
      Vs ← ∅
      As ← ∅
      Vs ← Vc
      As ← Ac
    end if
  else
    Vs ← Vc
    As ← Ac
  end if
end if
if {|Vs| ≠ 0 ∧ |Vc| >= |Vs|} then
  return
end if
Ap ← a ∈ A : init(a) = c
if {Ap = ∅} then
  return
end if
for all {a ∈ Ap} do
  n ← v ∈ V : ter(a) = v
  Ac ← a
  if {n = r} then
    return
  end if
  Vc', Ac' ← Vc, Ac
  exploraCamino(r, n, Vc', Ac', Vs, As)
end for
return
```

ya hayan sido seleccionados como parte de los caminos mínimos. En el caso de que exista un ancestro común, se creará un arco desde los nodos correspondientes, pero en ningún caso se añadirán nodos adicionales. Esta decisión la tomamos basándonos en la posibilidad de que nos encontremos en una zona de alta conectividad, y este proceso iterativo pueda conducirnos a la inclusión de un número excesivo de nodos nuevos.

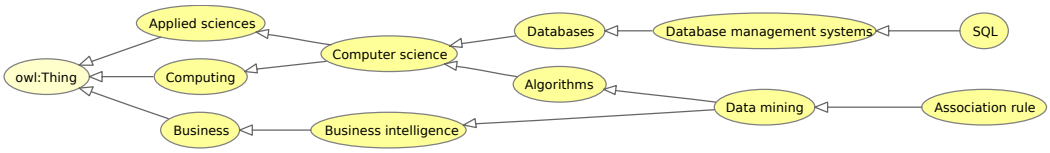


Figura 5.20: Resultado ideal para el método básico de camino mínimo

Como podemos ver este método tiene el inconveniente de eliminar los caminos de mayor longitud, ya que puede que estos contengan información relevante. En el caso del término *SQL* consideramos altamente relevante la rama que incluye conceptos relacionados con lenguajes de programación y gestión de datos, pero sin embargo esta se desestima. Teniendo en cuenta estos inconvenientes y los vistos anteriormente para el caso en el que se indica la longitud total del camino a recorrer, podemos ver que el uso de criterios puramente taxonómicos, puede no ser lo suficiente discriminante para llevar a cabo la tarea que deseamos realizar.

Método Taxonómico Combinado

Para evitar en cierta medida los inconvenientes de ambas aproximaciones, podemos realizar un método combinado que saque el máximo provecho a cada aproximación. Como ya hemos comentado con anterioridad, el principal problema que nos encontramos es que cuanto mas nos acercamos a la raíz del grafo de categorías, mas conectados están los nodos y más generales son los contenidos que presentan. Por otra parte, las categorías adyacentes a las categorías iniciales seleccionadas para los términos contienen mucha información relevante, y por tanto es interesante incluirlas, el problema es que cuanto mas nos alejamos, menos relevantes son y al estar mas conectadas, aportan mucha información general no relevante.

La propuesta que realizamos en este apartado consiste en realizar una primera fase con longitud de camino, en la que vamos a obtener todas las categorías cercanas relevantes, y una vez explorado el grafo a la distancia correspondiente, utilizar el método de caminos mínimos para cada una de las categorías que quedan sin ancestro asignado. De este modo evitamos la generalidad y la alta conectividad de las zonas cercanas a la raíz empleando un camino directo, mientras permitimos la ramificación y exploración de las zonas cercanas a las categorías de partida.

En la Figura 5.21 podemos ver como actúa el algoritmo en el grafo de ejemplo. Tenemos una primera fase que es igual a aplicar el método de longitud de camino de la Sección 5.2.1. La diferencia reside en que una vez explorada la longitud indicada, en lugar de unir los nodos huerfanos a la raíz con nuevos arcos creados a tal efecto, tal y como se puede ver en

la Figura 5.8, se realiza una búsqueda de camino mínimo hacia la raíz. La primera etapa de búsqueda con longitud de camino se observa en la Figura 5.21 marcada en colo rojo, mientras que la segunda etapa de búsqueda de caminos mínimos está resaltada en azul. Como siempre, los nodos coloreados representan los nodos iniciales, y el nodo relleno en negro representa la raíz del grafo dirigido.

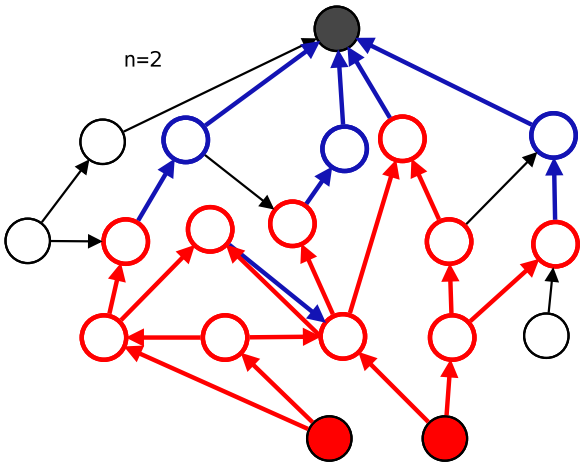


Figura 5.21: Método de exploración del grafo Taxonómico Combinado

En la Figura 5.22 podemos ver los resultados obtenidos para este método, con un valor de $n = 3$ para los términos *SQL* y *Association Rule*

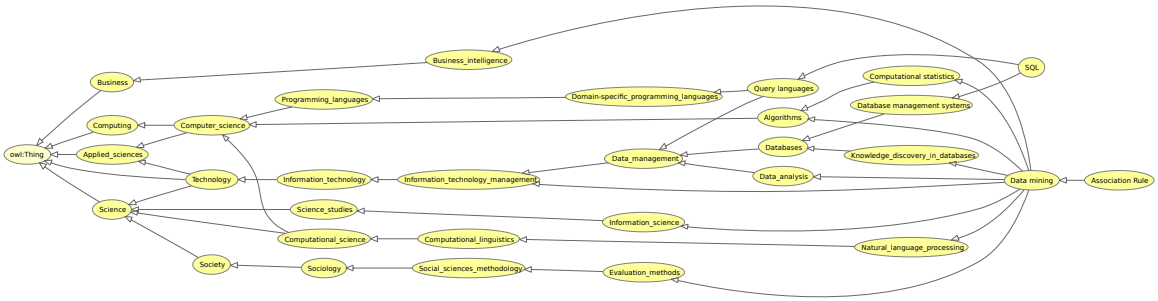


Figura 5.22: Resultado para el Método Taxonómico Combinado

La definición formal de este proceso se encuentra en el Algoritmo 10, este algoritmo realiza una exploración de longitud n y posteriormente aplica el procedimiento de búsqueda de caminos mínimos descrito en el Algoritmo 9.

El inconveniente de esta aproximación es que tanto en la etapa de exploración inicial, como en la segunda de caminos mínimos, se pueden incluir categorías que no sean relevantes o no estén completamente relacionadas con los términos que se están procesando. Para evitar este inconveniente lo mejor es evaluar cada una de las categorías a seleccionar y determinar si son o no relevantes.

Algoritmo 10 Algoritmo para el método Taxonómico Combinado

Input: El grafo $G_W = (V, A, r)$, los nodos iniciales $Vc = \{v_1, \ldots v_n\}$ y una longitud n
Output: El conjunto de nodos Vs y el conjunto de arcos As

$Vs_0 \leftarrow \emptyset, As_0 \leftarrow \emptyset$
$Vs_1 \leftarrow Vc, As_1 = \emptyset$
for $k \leftarrow 2$ **to** $n+1$ **do**
  for all $\{v_i \in Vs_{k-1} \wedge v_i \notin \bigcup_{j=0}^{k-2} Vs_j\}$ **do**
    if $\{a \in A, v \in V : init(a) = v_i \wedge ter(a) = v\}$ **then**
      $As_k \leftarrow a, Vs_k \leftarrow v$
    end if
    $Vs_k \leftarrow [Vs_k \backslash Vs_{k-1}]$
  end for
end for
for all $\{v_i \in Vs_k\}$ **do**
  $Vp, Ap \leftarrow \emptyset$
  $Vm, Am \leftarrow \emptyset$
  $exploraCamino(r, v_i, Vp, Ap, Vm, Am)$
  $Vs_{k+1} \leftarrow Vm$
  $As_{k+1} \leftarrow Am$
end for
$As \leftarrow \bigcup_{i=1}^{n+1} As_i, Vs \leftarrow \bigcup_{i=1}^{n+1} Vs_i$
return As, Vs

5.2.2. Métodos Basados en Medidas Semánticas

Como hemos visto, los métodos básicos nos permiten construir una ontología de salida de forma rápida. La validez de dicha ontología viene respaldada por el trabajo realizado por miles de usuarios de forma consensuada sobre el grafo de categorías de Wikipedia. Sin embargo, nuestro objetivo es obtener una ontología compacta y lo más representativa posible del dominio definido por los términos incluidos en la Estructura-AP.

Para conseguir esto, lo que debemos hacer es seleccionar un subárbol del grafo de categorías de Wikipedia, que sea lo bastante específico y lo más descriptivo posible. Para ello, debemos evaluar cada una de las categorías a incluir en la ontología, con el fin de determinar si son apropiadas o no.

Hasta ahora, hemos visto como los métodos exploratorios basados únicamente en la topología no podían garantizar realmente la relevancia de las categorías seleccionadas en cada caso. Mediante la exploración del grafo subyacente a Wikipedia, obtenemos información valiosa que está está implícita en la propia estructura del grafo, pero estamos obviando información muy importante como es el contenido de las páginas asociadas a cada una de las categorías.

Empleando medidas de distinta índole podemos obtener información relativa a la estructura del grafo y a su contenido. Una adecuada mezcla de esta medidas permitirá establecer diferentes criterios para la selección de las categorías adecuadas para incluirlas en la ontología.

A continuación vermos cada una de estos tipos de medidas.

5.2.3. Medidas con Índices Estadísticos

Dado que cada categoría contiene un conjunto de páginas, y una página está asociada a un conjunto de categorías, vamos a considerar cada categoría un conjunto, y las páginas muestras de ese conjunto. Teniendo en cuenta estas consideraciones, podemos aplicar medidas de tipo estadístico que nos proporcionen información relevante sobre el parecido de dos categorías en base a las páginas que contienen.

A continuación realizamos un breve repaso a distintas medidas estadísticas. Definiendo A y B como conjuntos de páginas de Wikipedia asociadas a las categorías a y b correspondientes.

Índice de similitud de Jaccard

El índice de similitud de Jaccard o coeficiente de similitud de Jaccard se emplea para estudiar la diversidad y similitud de un conjunto de muestras [Jaccard, 1901].

El índice de similitud de Jaccard se define como el tamaño de la intersección de las muestras dividido por el tamaño de la unión de las muestras:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}. \tag{5.1}$$

De forma análoga se define la distancia de Jaccard como el complemento del índice de similitud, y mide el grado de diversidad de un conjunto de muestras.

$$J_\delta(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}. \tag{5.2}$$

Índice de Sorensen-Dice

El índice de similitud de Sorensen o coeficiente de similitud de Sorensen, se emplea para determinar la similitud entre dos muestras [Sorensen, 1948].

$$QS = \frac{2|A \cap B|}{|A| + |B|}. \tag{5.3}$$

Este índice es equivalente al índice Dice [Dice, 1945], por lo que nos referimos a este índice como Sorensen-Dice.

Índice de Mountford

El índice de Mountford [Mountford, 1962], es un índice ligeramente dependiente del tamaño del conjunto de muestras.

$$M(A,B) = \frac{2|A \cap B|}{2|A||B| - (|A| + |B|)(|A \cap B|)}. \tag{5.4}$$

Medida de Solapamiento

La medida de solapamiento o índice de overlap, es una medida de similitud relacionada con el índice de Jaccard, que calcula el solapamiento entre dos conjuntos.

$$overlap(A,B) = \frac{|A \cap B|}{min(|A|,|B|)}. \tag{5.5}$$

Si A es un subconjunto de B o viceversa el índice da el valor 1.

Selección de Categorías usando Índices Estadísticos

Vamos a ver cómo la selección de categorías utilizando los criterios dados por los índices estadísticos, pueden ayudarnos a definir una ontología más compacta. La idea principal de estas medidas es la de emplearlas sobre conjuntos de elementos, ya que vemos que todas ellas se basan en el cálculo del cardinal de los conjuntos, de su unión o su intersección. Así pues el proceso que vamos a seguir es el siguiente:

- Por cada término que tengamos en la Estructura-AP y que verifique las reglas sintácticas básicas que ya vimos con anterioridad en la sección 4.4.5, obtenemos su página correspondiente de Wikipedia. En el caso de términos compuestos, si no encontramos la página, probamos con sus componentes, es decir, si no encontramos *Fuzzy Relational Database* buscaremos *Fuzzy Database* y *Relational Database* (pero siempre seleccionando términos que verifiquen las reglas sintácticas básicas). En el caso básico se generaba una ontología de inclusión, ya que no teníamos herramientas

externas en las que apoyarnos, sin embargo, en este caso sólo debe preocuparnos encontrar los términos adecuados a partir de los cuales comenzar la exploración del grafo de categorías.

- Seleccionamos la categoría o categorías a las que pertenece la página correspondiente con el término y a partir de ellas exploramos las categorías ancestro vecinas en el grafo.

- Exploramos el grafo y vamos comparando las categorías que encontramos en el camino. Como veremos más adelante la decisión acerca de con cual categoría comparar la categoría evaluada actualmente puede variar, pero en cualquier caso, fruto de esta comparación mediante los índices estadísticos vistos anteriormente, debemos obtener un valor. El valor calculado con el índice correspondiente debe ser mayor al umbral t que se haya fijado previamente. Cada categoría evaluada cuyo índice sea superior al umbral se incluirá en el resultado final.

- El proceso es el mismo para cada término, una vez terminado el proceso para todos los términos, se unen todas las categorías que se han seleccionado.

Dado que el uso de los índices presentados está orientado a conjuntos, tomaremos cada categoría como un conjunto de las páginas de Wikipedia categorizadas en ella. Para poder realizar la comparación de dos categorías, consideraremos cada página representada por su título. En este caso tratamos los títulos como un elemento completo, no se buscarán términos incluidos en los títulos, sino los títulos íntegros. En la Figura 5.23 vemos un ejemplo de comparación entre dos categorías C_1 y C_2, donde cada una tiene un conjunto de páginas $C_1 = \{p_a, p_b, p_c\}$ y $C_2 = \{p_b, p_d\}$. Cada página tiene un título, el conjunto de los títulos de las páginas en C_1 se denomina A y el de los títulos de las páginas de C_2 se denomina B. La aplicación de un índice estadístico sobre dos conjuntos devuelve el valor de similitud entre dichos conjuntos y por tanto entre las categorías.

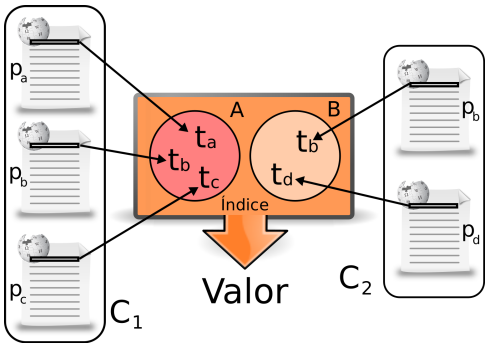


Figura 5.23: Comparación de categorías usando índices estadísticos

Un elemento clave es determinar el marco de referencia en el que vamos a realizar la comparación, en los apartados siguientes veremos distintas propuestas.

Usando la categoría inicial

Este método es el más básico, establecemos como categoría de referencia la categoría original desde la que se inicia la exploración del grafo. Conforme recorremos el grafo con el método básico de exploración visto en la sección 5.2.1 vamos comparando las categorías visitadas con la original. Si se supera el umbral t predefinido, se incluye a la nueva categoría en el resultado final, si no es así, esta no se incluye y no se continúa la expansión a partir de ella. Hemos elegido el método básico de exploración ya que las otra variantes eliminan nodos intermedios que desde el punto de vista de su contenido podrían ser relevantes.

Podemos ver el proceso de exploración y comparación de categorías en el Algoritmo 11. Debemos destacar el uso que se hace de las funciones *category*() que devuelve la categoría asociada a un vertice del grafo, *pages*() que devuelve las páginas de una categoría y *title*() que devuelve el título de una o varias páginas. También tenemos la función *compare*() que devuelve la similitud dados dos conjuntos, esta función puede ajustarse a cualquiera de los índices presentados anteriormente haciendo referencia a ella como $compare_{Jaccard}()$, $compare_{Mountford}()$, etc...

Algoritmo 11 Algoritmo de Exploración Básico con comparación con la Categoría Inicial

Input: El grafo $G_W = (V, A, r)$, un conjunto de nodos iniciales $Vc = \{v_1, \ldots, v_n\}$ y un umbral t

Output: El conjunto de nodos Vs y el conjunto de arcos As

```
for all v_l ∈ Vc do
    C_1 ← title(pages(category(v_l)))
    Vs_0 ← ∅, As_0 ← ∅
    Vs_1 ← v_l, As_1 ← ∅
    k ← 2
    for all {v_i ∈ Vs_{k-1} ∧ v_i ∉ ⋃_{j=0}^{k-2} Vs_j} do
        if {a ∈ A, v ∈ V : init(a) = v_i ∧ ter(a) = v} then
            C_2 ← title(pages(category(v)))
            if compare(C_1, C_2) >= t then
                As_k ← a, Vs_k ← v
            end if
        end if
        Vs_k ← [Vs_k \ Vs_{k-1}]
        k++
    end for
    As ← ⋃_{i=1}^{k-1} As_i, Vs ← ⋃_{i=1}^{k-1} Vs_i
end for
return As, Vs
```

En la figura 5.24 mostramos el subconjunto del grafo en el que se incluyen todos los caminos desde el término *SQL* que superan un umbral $t = 0.0001$, al comparar con la

categoría inicial *SQL*.

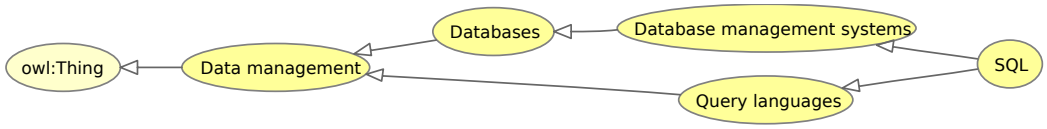


Figura 5.24: Resultado para las métricas de similitud

En la Tabla 5.2 vemos los resultados de cada métrica de similitud para dos pares de categorías C_1, C_2.

Tabla 5.2: Valores de similitud usando distintos índices, comparando con la categoría inicial

C_1	C_2	Jaccard	Sorensen	Mountford	Solapamiento
SQL	Database management systems	0.077777	0.144329	0.001898	0.197183
SQL	Query languages	0.061403	0.115702	0.002239	0.14
SQL	Databases	0.015384	0.030303	0.000304	0.056338
SQL	Data management	0.015384	0.030303	0.000285	0.070423

Atendiendo a los resultados, podemos ver que independientemente de la métrica utilizada, el conjunto de categorías seleccionadas es el mismo. Vemos como, conforme nos alejamos de la categoría inicial, los valores de similitud van disminuyendo, reforzando la idea de que a mayor distancia topológica menor es la similitud entre categorías.

Como podemos ver, este método tiene el inconveniente de que al establecer un único marco de referencia posible para las comparaciones, conforme ascendemos en el grafo, las categorías cada vez tendrán menos relación con la categoría original. Sin embargo sí que puede existir una relación por transitividad, y por esto planteamos el método que mostramos a continuación.

Usando la categoría actual

En esta propuesta iremos variando el marco de referencia, de tal modo que la categoría con la que vamos a establecer la comparación es con la última que se ha aceptado. Mantendremos una cola de categorías aceptadas, y las iremos expandiendo, de tal forma que una categoría se compara con todas sus categorías padre. Esto permite ir seleccionado un nuevo contexto en cada momento y en base a él seleccionar categorías próximas.

El proceso seguido se encuentra descrito formalmente en el Algoritmo 12.

El resultado obtenido se muestra en la Figura 5.25, vemos entonces que el paso desde *Query Languages* o *Data Management* a otra categoría superior implica aumentar la generalidad y por tanto no existe relación entre los términos empleados.

En la Tabla 5.3 mostramos los valores obtenidos para los distintos índices al comparar la categoría actual con sus categorías padre.

De nuevo podemos observar que conforme ascendemos en la jerarquía las categorías son más distintas entre sí. Dos categorías cercanas entre sí en una zona cercana a los extremos

Algoritmo 12 Algoritmo de Exploración Básico con comparación con la Categoría Actual

Input: El grafo $G_W = (V, A, r)$, un conjunto de nodos iniciales $Vc = \{v_1, \dots v_n\}$ y un umbral t

Output: El conjunto de nodos Vs y el conjunto de arcos As

for all $v_l \in Vc$ **do**
  $Vs_0 \leftarrow \emptyset, As_0 \leftarrow \emptyset$
  $Vs_1 \leftarrow v_l, As_1 \leftarrow \emptyset$
  $k \leftarrow 2$
  for all $\{v_i \in Vs_{k-1} \wedge v_i \notin \bigcup_{j=0}^{k-2} Vs_j\}$ **do**
    if $\{a \in A, v \in V : init(a) = v_i \wedge ter(a) = v\}$ **then**
      $C_1 \leftarrow title(pages(category(v_i)))$
      $C_2 \leftarrow title(pages(category(v)))$
      if $compare(C_1, C_2) >= t$ **then**
        $As_k \leftarrow a, Vs_k \leftarrow v$
      end if
    end if
    $Vs_k \leftarrow [Vs_k \backslash Vs_{k-1}]$
    $k++$
  end for
  $As \leftarrow \bigcup_{i=1}^{k-1} As_i, Vs \leftarrow \bigcup_{i=1}^{k-1} Vs_i$
end for
return As, Vs

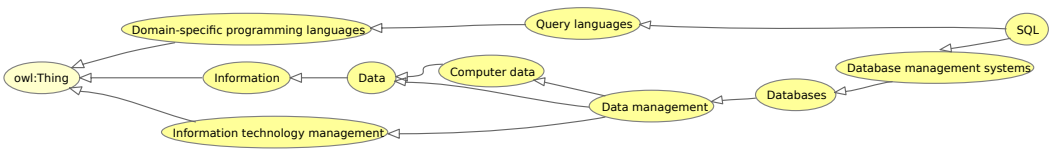


Figura 5.25: Resultado para las métricas de similitud comparando con la categoría actual

Tabla 5.3: Valores de similitud usando distintos índices, comparando con la categoría actual

C_1	C_2	Jaccard	Sorensen	Mountford	Solapamiento
SQL	SQL	1	1	1/0	1
SQL	Database management systems	0.077778	0.14433	0.001898	0.197183
SQL	Query languages	0.061404	0.115702	0.002239	0.14
Database management systems	Databases	0.071186	0.132911	0.001028	0.170732
Query languages	Data management	0.003247	0.006472	0.000078	0.02
Query languages	Domain-specific progr. lang.	0.016393	0.032258	0.000559	0.04
Databases	Data management	0.046296	0.088496	0.00044	0.103627
Databases	Computer data	0.004065	0.008097	0.000097	0.018519
Databases	Computer science	0	0	0	0
Domain-specific progr. lang.	Programming languages	0	0	0	0
Data management	Project management	0	0	0	0
Data management	Information tech. management	0.016162	0.031809	0.000131	0.032787
Data management	Information retrieval	0	0	0	0
Data management	Computer data	0.029605	0.057508	0.000716	0.166667
Data management	Data	0.007435	0.01476	0.000705	0.166667
Information tech. management	Information technology	0	0	0	0
Information tech. management	Project management	0.006479	0.012876	0.000056	0.013514
Computer data	Data	0.03125	0.060606	0.003436	0.166667
Computer data	Computing	0	0	0	0
Data	Information	0.02439	0.047619	0.00295	0.083333
Information	Concepts	0	0	0	0

inferiores del grafo son más similares que dos que se encuentren a la misma distancia pero estén situadas en una zona cercana a la raíz del grafo.

El problema que nos encontramos en esta ocasión, es que si bien la expansión local se realiza de forma adecuada, cuando llegamos a categorías más generales y abstractas, por lo general estas no se suelen aceptar. También hemos de tener en cuenta que realizamos una búsqueda localizada, con respecto al contexto actual, pero estamos ignorando por completo el resto de categorías que ya hemos aceptado y que por tanto dotan de significancia semántica al resultado final. Para evitar convertir el proceso en una expansión localizada en la siguiente sección proponemos un nuevo método.

Usando las categorías acumuladas

Para convertir la inclusión de una nueva categoría en el resultado en una decisión a escala global en lugar de una decisión local, vamos a establecer como contexto de comparación un conjunto que es la unión de todas las categorías que se han aceptado. Ahora en lugar de comparar con una categoría concreta, comparamos con una categoría de referencia que es la unión de todas las categorías parciales que se han aceptado como parte del resultado final.

La descripción formal del proceso la podemos encontrar en el Algoritmo 13.

En la Tabla 5.4 se muestran los valores obtenidos para el término SQL con un umbral de 0.02, podemos ver cómo los valores sufren variaciones, no como en los dos casos anteriores donde la similitud iba decreciendo al acercarse a la raíz.

De forma general, podemos ver que los valores arrojados por las medidas de Jaccard y Mountford son muy bajos y no se corresponderían con una valoración realizada por un humano, el resto de medidas, a pesar de ser también bajas representan de forma más adecuada valores de similitud. El valor del umbral nos permitirá controlar el tamaño de la ontología final, siempre teniendo en cuenta que un umbral excesivamente bajo puede provocar la aceptación de demasiadas categorías en las zonas cercanas a la raíz y por tanto

Algoritmo 13 Algoritmo de Exploración Básico, estableciendo comparación con las Categoría Acumuladas

Input: El grafo $G_W = (V, A, r)$, un conjunto de nodos iniciales $Vc = \{v_1, \ldots v_n\}$ y un umbral t

Output: El conjunto de nodos Vs y el conjunto de arcos As

for all $v_l \in Vc$ **do**
  $C_1 \leftarrow title(pages(category(v_l)))$
  $Vs_0 \leftarrow \emptyset, As_0 \leftarrow \emptyset$
  $Vs_1 \leftarrow v_l, As_1 \leftarrow \emptyset$
  $k \leftarrow 2$
  for all $\{v_i \in Vs_{k-1} \wedge v_i \notin \bigcup_{j=0}^{k-2} Vs_j\}$ **do**
    if $\{a \in A, v \in V : init(a) = v_i \wedge ter(a) = v\}$ **then**
      $C_2 \leftarrow title(pages(category(v)))$
      if $compare(C_1, C_2) >= t$ **then**
        $As_k \leftarrow a, Vs_k \leftarrow v$
        $C_1 \leftarrow C_1 \cup C_2$
      end if
    end if
    $Vs_k \leftarrow [Vs_k \backslash Vs_{k-1}]$
    $k++$
  end for
  $As \leftarrow \bigcup_{i=1}^{k-1} As_i, Vs \leftarrow \bigcup_{i=1}^{k-1} Vs_i$
end for
return As, Vs

Tabla 5.4: Valores de similitud usando distintos índices, comparando con la categoría acumulada

C_1	Jaccard	Sorensen	Mountford	Solapamiento
Query languages	0.061404	0.115702	0.002239	0.14
Database management systems	0.077273	0.14346	0.001416	0.149123
Data management	0.027897	0.05428	0.000241	0.059091
Domain-specific progr. lang.	0.003717	0.007407	0.000059	0.027027
Databases	0.065598	0.123119	0.000515	0.233161
Information technology management	0.01087	0.021505	0.000061	0.040984
Project management	0.003515	0.007005	0.00002	0.018018
Computer data	0.010373	0.020534	0.000223	0.185185
Information retrieval	0	0	0	0
Data	0.002053	0.004098	0.000189	0.166667
Programming languages	0	0	0	0
Computer science	0	0	0	0
Project management	0.003356	0.006689	0.000019	0.018018
Information technology	0.000995	0.001988	0.000033	0.03125
Computing	0	0	0	0
Information	0.00388	0.007729	0.000142	0.133333
Concepts	0	0	0	0

una ontología final de gran tamaño y con una gran cantidad de clases no relevantes.

Para este método podemos ver cómo la acumulación de las categorías aceptadas en un principio tiene un buen comportamiento y nos permite salir de zonas localizadas muy relacionadas para dar el salto a otras categorías generales. Sin embargo, es en este punto cuando la aproximación tiene su mayor inconveniente. Dado que la zona superior del grafo de Wikipedia está altamente conectada, una vez que aceptamos una categoría general esto nos va a llevar a aceptar otras, y así de forma iterativa podemos acabar incluyendo prácticamente el grafo completo. Para evitar esto podemos ajustar el valor del umbral. Por lo general los valores de similitud devueltos por la medidas son muy bajos, por lo que hay que escoger el umbral con mucho cuidado.

En este punto vemos cómo estos índices no nos proporcionan capacidad discriminatoria suficiente para obtener los resultados que buscamos. Por esto, vamos a analizar otro conjunto de medidas que teniendo en cuenta la similitud entre el contenido de las categorías y la topología, nos permitan realizar una selección adecuada.

5.2.4. Medidas de Similitud Semántica sobre Ontologías

Las medidas de similitud semántica hacen referencia a la proximidad o parecido entre dos conceptos dentro de una misma ontología. La distancia entre dos conceptos en una ontología viene dada por una representación numérica que va a indicar cómo de alejados están dos conceptos en un espacio geométrico determinado. Cuanto más similares o próximos son dos conceptos menor es la distancia entre ellos. Si la relación entre distancia y similitud se mantiene, se pueden usar las medidas para buscar elementos relacionados o asociaciones entre conceptos que no son reconocibles a simple vista.

Distinguiremos entre similitud semántica, que indica cierto grado de igualdad y viene dado por relaciones de sinonimia e hiperonimia, y relación semántica que es más general e incluye cualquier tipo de asociación léxica o funcional que pueda existir entre dos palabras (incluyendo la similitud). En la mayoría de aplicaciones es suficiente con tener información sobre relación semántica, pero es importante distinguir bien entre estos conceptos.

Un medida de similitud semántica toma como entrada dos conceptos y devuelve una valoración numérica que cuantifica cómo de parecidos son ambos conceptos. Estas medidas suelen basarse en relaciones jerárquicas definidas en una ontología o taxonomía a la cual pertenecen los conceptos evaluados.

Las medidas que utilizan ontologías para derivar medidas de similitud semántica suelen basarse en estrategias de tipo topológico, que miden la longitud del camino entre conceptos, o que ponderan los tramos del camino atendiendo a criterios de contenido de la información.

A continuación vamos a hacer un breve repaso a las medidas de similitud semántica más importantes. Las distinguiremos en dos grupos, medidas topológicas o de distancia sobre el conocimiento codificado en la ontología, que veremos en este apartado, y medidas de información que veremos en el siguiente.

Medida de la longitud del camino (*path-lenght*) de Rada et al.

La medida de la longitud del camino [Rada et al., 1989] calcula la similitud entre dos conceptos contando el número de nodos en el camino más corto entre ellos en una jerarquía de tipo *es-un* (*IS-A*). Matemáticamente la similitud entre dos conceptos c_1 y c_2 utilizando la medida de la longitud del camino se define como:

$$sim_{path}(c_1, c_2) = 1/p, \tag{5.6}$$

donde p es el número de nodos en el camino más corto entre los dos conceptos en la jerarquía. Esta medida tiene el principal inconveniente de que no tiene en cuenta el tamaño global de la jerarquía en la que se está evaluando los conceptos, la distancia y por tanto la similitud debe evaluarse en el contexto global de la jerarquía. Esa es la principal aportación de la siguiente medida.

Medida de Leacock-Chodorow

La medida de Leacock-Chodorow [Leacock and Chodorow, 1998] define la similitud entre dos conceptos c_1 y c_2 de una jerarquía como

$$sim_{lch}(c_1, c_2) = -log\left(\frac{p}{2 \cdot depth}\right), \tag{5.7}$$

donde p es el número de nodos en el camino más corto entre ambos conceptos en la jerarquía y *depth* es la profundidad máxima de la jerarquía.

Medida de Wu-Palmer

La medida de Wu-Palmer [Wu and Palmer, 1994] es una medida escalada que tiene en cuenta tanto la profundidad a la que se encuentran los nodos como la profundidad a la que se encuentra el ancestro común más cercano (*lowest common subsumer, lcs*).

$$sim_{wp}(c_1, c_2) = \frac{2 \cdot N3}{N1 + N2 + 2 \cdot N3}, \tag{5.8}$$

siendo c_3 el superconcepto común más cercano de c_1 y c_2, $N1$ es el número de nodos en el camino de c_1 a c_3, $N2$ es el número de nodos en el camino de c_2 a c_3 y $N3$ es el número de nodos en el camino desde c_3 a la raíz.

Posteriormente se realizó una reformulación en [Resnik, 1999], simplificando la original propuesta por los autores y teniendo en cuenta la profundidad de los nodos (*depth()*):

$$sim_{wp}(c_1, c_2) = \frac{2 \cdot depth(lcs_{c_1,c_2})}{depth(c_1) + depth(c_2)}. \tag{5.9}$$

Selección de Categorías empleando Medidas de Similitud Semántica en Ontologías

Aunque las medidas presentadas en los apartados anteriores no son de directa utilidad, puesto que en cierta forma ya las hemos usado en nuestras primeras aproximaciones de carácter intuitivo, es conveniente presentarlas y ofrecer una breve discusión acerca de ellas y el porqué no es adecuado emplearlas en este contexto.

Todas las medidas anteriores, son medidas de similitud semántica, y por tanto se apoyan en la estructura de hiperónimos/hipónimos del grafo. Si bien estas medidas nos indican la similitud entre dos nodos cualesquiera del grafo atendiendo a condiciones meramente topológicas, la aproximación elegida en nuestro caso en cierta forma las hace redundantes. Estas medidas son de gran utilidad para el cálculo de la similitud entre dos nodos cualesquiera de un grafo. Si comparásemos todos los nodos del grafo, podríamos seleccionar aquellos similares a los que nos interesan. Sin embargo, hemos desestimado esta forma de proceder debido a la gran cantidad de nodos que tiene el grafo de Wikipedia, y el alto coste computacional que ello conllevaría.

Así pues, nuestra decisión de ir seleccionado/podando el grafo a través de la navegación de este, ya lleva implícita la aplicación de una condición de similitud topológica. Cuando exploramos los ancestros directos de un nodo, damos por hecho que son muy similares de forma implícita y que esto se refleja en esa misma conexión. Si aplicamos la medida de Rada et al. a un nodo con sus ancestros, el valor de similitud obtenido es de 0.5 (el valor más alto posible para un nodo que no sea comparado consigo mismo). Si aplicásemos esta medida siguiendo un criterio de navegación de comparación con el nodo inicial, el uso de Rada sería equivalente a nuestra aproximación inicial de método básico con longitud de camino visto en la Sección 5.2.1. En este caso el parámetro n sería elegido a partir de un umbral t tal que $t = sim_{path}(c_1, c_2) = 1/n$, de modo que seleccionamos todos los nodos que comparados con el actual tenga una similitud mayor o igual a t. Aplicado a la segunda aproximación donde la comparación se realiza con el nodo inmediatamente anterior, la similitud siempre sería máxima, y nos encontraríamos con nuestro caso inicial de método básico visto en la Sección 5.2.1.

En el caso de la medida de Leacock-Chodorow nos encontramos con la misma situación, la similitud será muy alta entre dos nodos/categorías adyacentes, ya que el parámetro de profundidad es fijo en todos los casos. Dadas las características del grafo de categorías podíamos reemplazar el valor de profundidad de la jerarquía, por el valor del diámetro del grafo. La inclusión del parámetro profundidad aporta una mejora con respecto a Rada et al. en cuanto que la similitud se calcula en base a un contexto que es la jerarquía, pero en nuestro caso no aporta información relevante. En ambas formas de exploración, comparando con el nodo original y comparando con el nodo actual, la situación es equivalente a la presentada con respecto a Rada et al.

La última medida, la de Wu-Palmer, se adapta perfectamente a nuestra forma de recorrer el grafo, ya que se tiene en cuenta la profundidad de los nodos a comparar, no la profundidad total de la jerarquía como en Leacock-Chodorow, y sus ancestros comunes. Teniendo en cuenta que la comparación la hacemos con ancestros a los que llegamos por caminos sin ciclos, el menor ancestro común será uno de los nodos a comparar, por lo que

la medida en nuestro caso quedaría de la siguiente forma:

$$sim_{wp}(c_1, c_2) = \frac{2 \cdot depth(c_2)}{depth(c_1) + depth(c_2)}. \tag{5.10}$$

Tomando esta medida para el término *SQL* sobre las categorías obtenidas para el método básico de camino con $n = 4$ obtenemos los valores mostrados en la Figura 5.26. El cálculo se realiza con los valores de profundidad mínima para cada nodo en el grafo de categorías de Wikipedia. Cada nodo está etiquetado con su profundidad mínima en Wikipedia, que no tiene porqué corresponderse con la profundidad mostrada en el resultado final.

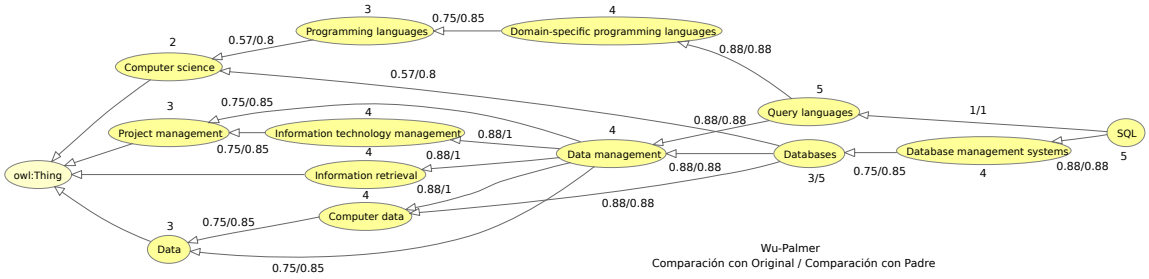


Figura 5.26: Valores de Wu-Palmer para el término *SQL* con n=4

Podemos observar cómo los valores obtenidos por la medida de Wu-Palmer son muy elevados comparados con las medidas de similitud vistas anteriormente. Al ser esta medida de naturaleza estrictamente taxonómica, queda incluida de forma implícita en la forma en que recorremos el grafo de categorías. El único inconveniente es que esta medida está diseñada para ser empleada en árboles, y al utilizarla en grafos como el de categorías de Wikipedia pueden surgir algunos inconvenientes. Si observamos detenidamente la Figura 5.26 vemos como el nodo correspondiente a *Databases* tiene dos valores para profundidad 3 y 5. Como comentamos anteriormente siempre emplearemos los valores de profundidad mínima, para evitar los ciclos y caminos poco informativos en el grafo. Por ejemplo, en esta ocasión si empleamos la profundidad mínima que es 3, al ser esta menor que la del antecesor común, en este caso *Data management* que tiene profundidad 4, el valor de similitud es mayor que 1, lo cual no tiene sentido. Esto se debe a que existen varios caminos hasta la raíz desde *Databases* y el que pasa por *Data management* no es mínimo. En estas circunstancias excepcionales se debe recalcular la similitud en base a la profundidad en el camino evaluado, en este caso sería 5, y ahora sí, es posible calcular la similitud.

Además según el criterio taxonómico se concedería el mismo valor a dos categorías topológicamente equivalentes con respecto a la categoría evaluada, sin tener el cuenta el contenido de las mismas. Es por esto, que consideramos conveniente buscar otras medidas que, aparte del componente topológico, tengan en cuenta el contenido textual de cada uno de los nodos tratados.

Con respecto a los valores obtenidos para cada una de las medidas con respecto a los diferentes métodos de comparación de categorías, en la Tabla 5.5 se muestran los valores para las distintas medidas de similitud semántica realizando una exploración básica y comparando con la categoría inicial.

Tabla 5.5: Valores de similitud usando medidas de similitud semántica, comparando con la categoría inicial

C_1	C_2	Leacock-Chodorow	Wu-Palmer	Path Length
SQL	SQL	2.995732	1	1
SQL	Query languages	2.302585	0.6	0.5
SQL	Database management systems	2.302585	0.888889	0.5
SQL	Data management	1.89712	0.666667	0.333333
SQL	Domain-specific progr. lang.	1.89712	0.6	0.333333
SQL	Databases	1.89712	0.666667	0.333333
SQL	Data	1.386294	0.6	0.2
SQL	Project management	1.609438	0.6	0.25
SQL	Computer data	1.609438	0.6	0.25
SQL	Information technology management	1.609438	0.666667	0.25
SQL	Information retrieval	1.386294	0	0.2
SQL	Programming languages	1.609438	0.666667	0.25
SQL	Computer science	1.609438	0.666667	0.25
SQL	Information	1.386294	0.6	0.2
SQL	Management	1.609438	0.666667	0.25
SQL	Product development	1.386294	0.545455	0.2
SQL	Production and manufacturing	1.203973	0	0.166667
SQL	Computing	1.609438	0.666667	0.25
SQL	Information technology	1.609438	0.666667	0.25
SQL	Formal sciences	1.386294	0.6	0.2
SQL	Applied sciences	1.386294	0	0.2
SQL	Information	1.386294	0.6	0.2
SQL	Information science	1.203973	0	0.166667
SQL	Computer languages	1.609438	0.6	0.25
SQL	Programming language topics	1.386294	0.6	0.2
SQL	Computer programming	1.386294	0.6	0.2
SQL	Digital technology	1.386294	0.6	0.2
SQL	Interdisciplinary fields	1.386294	0	0.2
SQL	Abstraction	1.203973	0.545455	0.166667
SQL	Scientific disciplines	1.203973	0	0.166667
SQL	Formalism (deductive)	1.203973	0.545455	0.166667
SQL	Academic disciplines	1.203973	0.545455	0.166667
SQL	Applied disciplines	1.203973	0.545455	0.166667
SQL	Concepts	1.609438	0.666667	0.25
SQL	Business	1.609438	0.666667	0.25
SQL	Business economics	1.386294	0.444444	0.2
SQL	Accountability	1.386294	0.4	0.2
SQL	Product management	1.203973	0.545455	0.166667
SQL	Industry	1.386294	0	0.2
SQL	Science studies	1.203973	0	0.166667
SQL	Library and information science	1.203973	0.545455	0.166667
SQL	Notation	1.386294	0.545455	0.2
SQL	Constructed languages	1.386294	0.6	0.2
SQL	Software engineering	1.609438	0.666667	0.25

Lo primero que conviene resaltar de los resultados obtenidos es que la medida de Leacock-Chodorow no es una medida normalizada. Podemos observar en la primera columna de resultados como la comparación de la categoría *SQL* consigo misma proporciona una valor de similitud de 2.995732. Esto se debe al escalado logarítmico que utiliza la medida. También podemos observar que la medida de Rada tiene una cota superior de 1, que corresponde al caso en que ambos nodos son el mismo, y que proporciona valores que tienden a 0 cuanto mayor es la distancia entre los dos nodos, aunque por definición nunca se alcanzará el valor 0. Con respecto a la medida de de Wu-Palmer, ésta proporciona un valor de 1, cuando ambos nodos son el mismo, y por tanto dicho nodo es a su vez *lcs* de sí mismo. El valor 0 se obtiene cuando los dos nodos no tienen nodos ancestros en común, y por tanto, su *lcs* es la raíz, que por definición tiene prorfundidad 0.

Los datos obtenidos confirman que aunque la caracterización topológica de los nodos sea apropiada, la semántica de ésta difiere bastante de unos casos a otros, y eso no lo reflejan estas medidas. Tomemos como ejemplo las categorías *Databases* y *Data Management*, ambas están a la misma profundidad y a la misma distancia de la categoría inicial *SQL*. Esto implica para Rada y Leacock-Chodorow, que ambas tienen una relación equivalente con respecto a la categoría inicial. Sin embargo, podemos ver que intuitivamente *Databases* está más relacionada con *SQL* de lo que lo está *Data Management*. Esta casuística varía en el caso de la medida de Wu-Palmer, ya que ésta obtiene el *lcs*, el cual puede diferir para distintas categorías situadas a la misma profundidad. Sin embargo, para el ejemplo que estamos empleando podemos ver que el *lcs* en ambos casos está a la misma profundidad (en ambos casos es *Data Management*) y por lo tanto perdedemos la capacidad discriminatoria entre distintas categorías, topológicamente equivalentes, pero semánticamente diferenciables.

Sin cambiamos el marco de referencia de la comparación y la vamos estableciendo entre clases hijas y padres conforme se recorre el grafo, obtenemos los valores mostrados en la Tabla 5.6.

Los datos obtenidos nos permiten poner de manifiesto de forma clara las observaciones realizadas anteriormente con respecto a la poca capacidad de discriminación de estas medidas. En esta ocasión el grafo se recorre como en el caso anterior de forma ascendente hacia la raíz, pero ahora comparamos cada nueva categoría a explorar con sus padres. La idea inicial de este tipo de método es la de determinar grandes saltos conceptuales en el camino hacia la raíz y detener en ese momento la exploración. Estos cambios drásticos se corresponden con la entrada en zonas superiores del grafo donde los conceptos son más genéricos. Sin embargo, dada la naturaleza de estas medidas, no es posible distinguir mas allá de la posición relativa de las categorías lo que provoca que los datos obtenidos sean prácticamente iguales en todos los casos.

En el caso de la medida de Rada, el valor será siempre de 0.5 puesto que la longitud entre un nodo hijo y su padre es 2 (recordemos que se cuentan los nodos en el camino, para evitar la división por cero en el caso en que se compare un nodo consigo mismo). En el caso de Leacock-Chodorow ocurre algo parecido. Dado que la profundidad total es un valor fijo, y el único valor que varía es la longitud entre nodos (que en este caso es fija), obtendríamos siempre el mismo valor para todos los nodos. Para la medida de Wu-Palmer observamos como existe algo más de variación atendiendo a localización del *lcs*. Aunque

Tabla 5.6: Valores de similitud usando medidas de similitud semántica, comparando con la categoría actual

C_1	C_2	Leacock-Chodorow	Wu-Palmer	Path Length
SQL	SQL	2.995732	1	1
SQL	Query languages	2.302585	0,6	0,5
SQL	Database management systems	2.302585	0,888889	0,5
Database management systems	Databases	2.302585	0,75	0,5
Query languages	Data management	2.302585	0,888889	0,5
Query languages	Domain-specific progr. lang.	2.302585	0,6	0,5
Databases	Computer data	2.302585	0,666667	0,5
Databases	Computer science	2.302585	0,75	0,5
Databases	Data management	2.302585	0,75	0,5
Data management	Information retrieval	2.302585	0	0,5
Data management	Data	2.302585	0,666667!	0,5
Data management	Project management	2.302585	0,666667	0,5
Data management	Computer data	2.302585	0,666667!	0,5
Data management	Information technology management	2.302585	0,75	0,5
Domain-specific progr. lang.	Programming languages	2.302585	0,888889	0,5
Computer data	Computing	2.302585	0,888889	0,5
Computer data	Data	2.302585	0,6	0,5
Computer science	Formal sciences	2.302585	0,666667	0,5
Computer science	Applied sciences	2.302585	0	0,5
Computer science	Computing	2.302585	0,75	0,5
Information retrieval	Information science	2.302585	0,888889	0,5
Data	Information	2.302585	0,888889	0,5
Project management	Production and manufacturing	2.302585	0	0,5
Project management	Management	2.302585	0,666667!	0,5
Project management	Product development	2.302585	0,909091	0,5
Information technology management	Project management	2.302585	0,666667	0,5
Information technology management	Information technology	2.302585	0,75	0,5
Programming languages	Programming language topics	2.302585	0,666667	0,5
Programming languages	Computer programming	2.302585	0,666667	0,5
Programming languages	Computer science	2.302585	0,75	0,5
Programming languages	Computer languages	2.302585	0,666667	0,5
Computing	Digital technology	2.302585	0,666667	0,5
Computing	Information technology	2.302585	0,75	0,5
Formal sciences	Formalism (deductive)	2.302585	0,909091	0,5
Formal sciences	Interdisciplinary fields	2.302585	0	0,5
Formal sciences	Academic disciplines	2.302585	0,545455!	0,5
Formal sciences	Abstraction	2.302585	0,545455!	0,5
Formal sciences	Scientific disciplines	2.302585	0	0,5
Applied sciences	Applied disciplines	2.302585	0	0,5
Applied sciences	Scientific disciplines	2.302585	0,857143	0,5
Information science	Information	2.302585	0	0,5
Information science	Science studies	2.302585	0,857143	0,5
Information science	Library and information science	2.302585	0	0,5
Information science	Interdisciplinary fields	2.302585	0,571429	0,5
Information	Concepts	2.302585	0,857143	0,5
Production and manufacturing	Industry	2.302585	0,888889	0,5
Management	Business	2.302585	0,75	0,5
Management	Accountability	2.302585	0,444444	0,5
Management	Business economics	2.302585	0,5	0,5
Management	Applied sciences	2.302585	0	0,5
Product development	Production and manufacturing	2.302585	0	0,5
Product development	Product management	2.302585	0,909091	0,5
Programming language topics	Software engineering	2.302585	0,888889	0,5
Programming language topics	Computer languages	2.302585	0,6	0,5
Computer programming	Software engineering	2.302585	0,666667	0,5
Computer programming	Computing	2.302585	0,888889	0,5
Computer languages	Constructed languages	2.302585	0,6	0,5
Computer languages	Notation	2.302585	0,545455	0,5
Computer languages	Computing	2.302585	0,888889	0,5

esta medida destaca en este grupo por ser más discriminante, realmente podemos ver que por sí misma no nos ofrece resultados especialmente relevantes.

Vemos claramente que este método de comparación entre padres e hijos en el camino a la raíz no es adecuado para emplearlo con las medidas de similitud semántica. Sin embargo, nos ha servido para poner de manifiesto claramente el poco poder discriminatorio que poseen estas medidas. Parece claro que si queremos caracterizar de forma adecuada la comparación entre categorías debemos tener en cuenta la semántica de su contenido.

5.2.5. Medidas de Contenido de la Información

La limitación de las medidas basadas únicamente en distancias entre nodos en una jerarquía, es que el grado de similitud que determina un enlace directo entre conceptos no es consistente, ya que puede variar según la semántica de los conceptos implicados. Los enlaces entre conceptos generales implican una similitud menor que los enlaces entre conceptos muy específicos, y en ambos casos la longitud es 1.

Para evitar este tipo de problemas aparecen una serie de técnicas que no sólo tienen en cuenta la topología de la jerarquía empleada, sino que también tienen en cuenta el contenido de la información intrínseco a cada concepto. A continuación vamos a ver algunas de dichas medidas.

Contenido de Información de Resnik

En [Resnik, 1995], se presenta una nueva medida de similitud semántica sobre una taxonomía *IS-A* basada en la idea de contenido de información. La medida de contenido de información de Resnik (*Resnik Information Content*), combina la estructura taxonómica con estimaciones de probabilidad para conceptos obtenidas empíricamente sobre grandes corpus textuales, proporcionando una forma de adaptar una estructura de conocimiento estática a múltiples contextos. La similitud se modela como el grado en que los conceptos comparten información y viene dado por el contenido de información de su ancestro común más cercano (lcs).

$$sim_{res}(c_1, c_2) = IC(lcs_{c_1,c_2}). \tag{5.11}$$

Sólo nos queda definir el valor de la función de contenido de información (IC). En el artículo original se define el contenido de la información en base a medidas de frecuencias obtenidas de corpus textuales.

$$IC(c) = -log\left(\frac{freq(c)}{freq(root)}\right), \tag{5.12}$$

donde $freq(c)$ es la frecuencia del concepto c y $freq(root)$ es la frecuencia de la raíz de la jerarquía. La frecuencia de un concepto en la taxonomía se obtiene a partir de la frecuencia calculada sobre el *Brown Corpus of American English*, un corpus que contiene más de un millón de palabras, conteniendo fragmentos de noticias o de obras de ficción. Es conveniente reseñar, que en el conteo de frecuencias un término anota en su concepto correspondiente y en todos aquellos que los subsumen.

En [Seco et al., 2004] se presenta una variante de la medida de contenido de información de Resnik, únicamente dependiente de WordNet que no necesita de corpus externos para extraer datos estadísticos.

$$IC_{wn}(c) = 1 - \frac{log(hypo(c) + 1)}{log(max_{wn})}, \tag{5.13}$$

donde la función $hypo(c)$ devuelve el número de hipónimos (subconceptos mas específicos) para el concepto c y max_{wn} es una constante cuyo valor es el número total de conceptos existentes en la taxonomía. Esta reformulación del contenido de la información es equivalente a la de Resnik si tomamos la frecuencia de cada término como 1. Ambas definiciones proporcionan valores similares, lo que indica que la información sobre las frecuencias no es tan importante como la información sobre la topología, particularmente sobre la profundidad a la que se encuentran los conceptos.

Medida de similitud de Lin

El coeficiente de información de Resnik tiene el inconveniente de que dado que puede existir una gran variedad de conceptos que compartan el mismo ancestro común (lcs), no hay forma de realizar distinciones entre estos, ya que la medida se realiza únicamente sobre éste. Para solventar este inconveniente en [Lin, 1998] se propone escalar el contenido de la información del ancestro común atendiendo a los valores de contenido de la información de los conceptos subsumidos por éste. Para ello, Lin emplea una ratio, calculando la similitud de dos conceptos c_1 y c_2 como:

$$related_{lin}(c_1, c_2) = \frac{2 \times IC(lcs(c_1, c_2))}{IC(c_1) + IC(c_2)}, \tag{5.14}$$

donde $lcs(c_1, c_2)$ es el menor ancestro común y IC devuelve el contenido de la información del concepto.

En [Patwardhan et al., 2003] se interpreta esta medida como el contenido de la información de la intersección de dos conceptos (multiplicados por 2) y divididos por la suma de sus contenidos de información, lo que es análogo el coeficiente de Sorensen-Dice visto en la Sección 5.2.3. Es más, nosotros añadimos que es una adaptación de la medida de Wu-Palmer redefinida por Resnik y vista en la sección 5.2.4, sólo que en lugar de emplear criterios topológicos para evaluar los nodos, la profundidad en este caso, emplea criterios de contenido de información de los nodos.

Distancia de Jiang y Conrath

Otra alternativa para eliminar la poca capacidad discriminatoria de la medida de Resnik fue propuesta en [Jiang and Conrath, 1997], en este caso Jiang y Conrath realizan el escalado del contenido de información con respecto a sus subconceptos por medio de una diferencia. Esta medida calcula la distancia semántica (inversa de la similitud) para los conceptos c_1 y c_2 como:

$$dist_{jcn}(c_1, c_2) = IC(c_1) + IC(c_2) - 2 \times IC(lcs(c_1, c_2)). \quad (5.15)$$

Para obtener la relación semántica normalizada se reformula como:

$$related_{jcn}(c_1, c_2) = \frac{2 - (IC(c_1) + IC(c_2) - 2 \times IC(lcs(c_1, c_2)))}{2}. \quad (5.16)$$

Medida de Hirst & St.Onge

La medida de Hirst & St.Onge [Hirst and St.Onge, 1998] se basa en WordNet y determina la relación entre dos conceptos atendiendo a la naturaleza del camino que los une. Para ello se clasificaron las relaciones entre *synsets* en WordNet en relaciones de dirección ascendente, descendente y horizontal. Para cada palabra, se tienen en cuenta todos los *synsets* para los cuales la palabra tenga el sentido adecuado. Atendiendo al camino entre *synsets*, un camino relativamente corto y con pocos cambios de dirección representa un alto grado de similitud comparado con un camino más largo y con muchos cambios de dirección.

A pesar de que esta medida fue definida teniendo en cuenta la estructura jerárquica y las relaciones adicionales de WordNet, la consideramos útil como punto de referencia a la hora de abordar la exploración del grafo de Wikipedia, por lo que vamos a describirla con más detalle.

La medida de Hirst & St.Onge define tres tipo de relaciones, *extra-fuertes*, *fuertes* y *medias-fuertes*.

Las relaciones *extra-fuertes* se dan entre una palabra y su repetición literal. Este tipo de relaciones tendrán el peso máximo.

Se consideran tres tipos de relaciones *fuertes*, cuando existe un *synset* común a dos palabras, cuando existe un enlace horizontal entre *synsets* de cada palabra (antonimia, similitud, etc...) y cuando una palabra contiene a otra y existe cualquier tipo de enlace entre sus *synsets*. Una relación *fuerte* tiene un peso menor que una *extra-fuerte* y un peso mayor que una *media-fuerte*.

Una relación *media-fuerte* entre dos palabras tiene lugar cuando un miembro de un conjunto de caminos permitidos conecta los *synsets* de cada palabra. Un camino se define como una secuencia de entre dos y cinco enlaces entre *synsets*, y es permitido si se corresponde con unos patrones determinados mostrados en [Hirst and St.Onge, 1998] y definidos por las siguientes reglas.

1. Un enlace ascendente no puede venir precedido por un cambio de dirección.
2. Se permite como máximo un cambio de dirección.
3. La regla anterior tiene una única excepción, se permite usar un enlace horizontal para realizar la transición de una dirección ascendente a una descendente.

Teniendo en cuenta que una dirección ascendente se corresponde con una generalización del concepto, mientras que una descendente se corresponde con una especificación. Las

direcciones horizontales son aquellas que aportan otra información sobre la relación que no sea de tipo jerárquico, tales como relaciones *parte_ de*, *similar_ a*, etc...

Al contrario que las relaciones fuertes y extra-fuertes las relaciones medias-fuertes pueden tener diferentes pesos. El peso de un camino se determina mediante la siguiente formula:

$$peso = C - longitud\_camino - k \cdot numero\_de\_cambios\_de\_direccion, \tag{5.17}$$

donde C y k son constantes. Por tanto, cuanto mayor sea la longitud del camino y más cambios de dirección tenga, menor será el peso y por tanto menor la similitud entre ambas palabras.

Debido a las características del grafo de Wikipedia y sobretodo a la multitud de ciclos que en él aparecen, consideramos que esta medida no se ajusta de manera adecuada a nuestros propósitos, pero consideramos que debía hacerse referencia a ella debido a los buenos datos obtenidos en su aplicación a WordNet.

Selección de Categorías empleando Medidas de Contenido de la Información

Las medidas basadas en contenido de la información tienen en cuenta tanto la estructura topológica de la jerarquía como la información sobre frecuencias de los términos extraídos de fuentes externas. Esta utilización de las frecuencias no es mas que una forma de establecer pesos que determinen la contribución de cada nodo al concepto global, entendiendo como tal en este caso la raíz de la jerarquía. Tal como queda demostrado con la medida de IC de Seco, la importancia de las frecuencias de los términos en la jerarquía no es tan determinante como la posición del nodo correspondiente al término en ésta.

Para poder calcular valores para estas medidas sobre Wikipedia hacemos uso del API JWPL [Zesch and Gurevych, 2010]. JWPL (Java Wikipedia Library) es una interfaz de programación de aplicaciones de código abierto que permite acceder a la información contenida en Wikipedia. Esta API nos permite construir una jerarquía a partir del grafo de categorías de Wikipedia, esta jerarquía será empleada posteriormente para el cálculo de los valores de la función $hypo(c)$ del contenido de información de Seco. El principal inconveniente para este conjunto de medidas como podemos ver, es la necesidad de adaptar el grafo a una estructura jerárquica, eliminando ciclos y caminos redundantes. La forma en que se implementa el grafo jerárquico es calculando los caminos de cada nodo correspondiente a una categoría hasta la raíz. Dada la gran cantidad de nodos y con el objetivo de mejorar la eficiencia se presupone que todos los subcaminos en un camino mínimo hacia la raíz también son mínimos para los nodos correspondientes. Esta heurística se aplica comenzando desde los nodos hoja hacia la raíz (recordemos que en la imagen de Wikipedia con la que estamos trabajando hay 381000 nodos hoja). Esto proporciona los caminos iniciales con mayor número de subcaminos. Pese a que el uso de esta heurística es necesaria para poder calcular la jerarquía, provoca un efecto no deseado. Puede darse la circunstancia en la que dos nodos tengan un *lcs* que no forme parte del camino mínimo entre dichos nodos, con lo cual dicha relación se perdería.

Las medidas de contenido de la información están pensadas para ser utilizadas sobre jerarquías, el uso sobre grafos con múltiples caminos al mismo nodo desvirtúa esta medida. Teniendo en cuenta la forma en que recorremos el grafo, de las hojas a la raíz, todo nodo que encontremos en el camino a la raíz será el *lcs* entre dicho nodo y el inicial. Así evitamos el calcular el *lcs*, dado que la jerarquía generada no representa de forma adecuada todas las relaciones. En este caso para Resnik tenemos que $sim_{res'}(c_1, c_2) = IC(c_2)$, ya que la forma de exploración del grafo garantiza que $lcs(c_1, c_2) = c_2$.

Las evaluaciones realizadas de las medidas se han realizado con el IC de Seco para evitar la necesidad de emplear corpus textuales externos. La exploración realizada es básica y se realiza la comparación con la categoría inicial.

Para la obtención de valores correctos para estas medidas es necesario poder obtener el IC de cada una de las categorías empleadas en el cálculo final. Para la medida de Resnik sólo necesitamos obtener el IC del *lcs*, mientras que para para Lin y Jiang-Conrath debemos conocer además el IC de c_1 y c_2, las categorías que se comparan. Para hacernos una idea de la contribución de cada uno de estos parámetros al resultado final incluimos los valores para estos en las tres primeras columnas. Tal y como hemos argumentado anteriormente dadas las características de la exploración que realizamos el *lcs* de c_1 y c_2 siempre es c_2. También hemos hecho referencia a la pérdida de relaciones sufrida al procesar el grafo mediante JWPL. En este caso un valor desconocido de IC para el *lcs* hace que no se pueda calcular la medida. Para ver cómo afecta esta circunstancia a las medidas, incluimos junto a cada una de ellas una versión en la que se considera que $lcs(c_1, c_2) = c_2$.

En la Tabla 5.7 mostramos los valores obtenidos para las medidas, realizando la comparación con la categoría inicial.

Lo primero que podemos observar en la tabla es que la heurística de generación de caminos a la raíz destruye muchas de las relaciones existentes en el grafo, como se puede observar ante la gran cantidad de valores de IC para los *lcs* que no se pueden calcular. Vemos que aplicando las medidas con la restricción de $lcs(c_1, c_2) = c_2$ se obtienen más valores pero tampoco es una diferencia significativa. Para realizar el análisis tendremos en cuenta las medidas realizadas según el criterio $lcs(c_1, c_2) = c_2$.

Llama la atención como la medida de Resnik ya proporciona algunos datos que van en contra de la intuición de estas medidas. Si comparamos la categoría *SQL* consigo misma obtenemos un valor de similitud de 0,8655, mientras que si la comparamos con *Formalism* obtenemos un valor de 0.976, siendo éste más general (y por tanto su IC debería ser menor). Además, el valor obtenido al comparar una categoría consigo misma es el IC de la categoría que es un valor fijo, cuando lo adecuado sería obtener un valor de 1.

En el caso de los datos obtenidos para la medida de Lin vemos que se obtiene para *Formalism* un valor de 1,0292, esto se debe a que al realizar la eliminación de ciclos esta categoría ha pasado a estar más cerca de un nodo hoja que *SQL*. Igual ocurre con la medida de Jiang-Conrath en la que debido a este fenómeno vemos como obtenemos valores mayores de 1.

Los resultados iniciales no son nada intuitivos, por lo que esta familia de medidas aplicadas a Wikipedia no tiene mucha utilidad.

Comprobaremos si la aplicación de estas medidas mejora al aplicar otro tipo de comparación. En la Tabla 5.8 se muestran los resultados obtenidos para las medidas de con-

Tabla 5.7: Valores de similitud usando medidas de contenido de información, comparando con la categoría inicial

C_1	C_2	IC(C_1)	IC(C_2)	IC(lcs)	Resnik	R.IC(C_2)	Lin	L.IC(C_2)	J-C	JC.IC(C_2)
SQL	SQL	0.8655	0.8655	0.8655	0.8655	0.8655	1	1	1	1
SQL	Query languages	0.8655	0.8201	0	0	0.8201	0	0.973	0.1572	0.9773
SQL	Database manag. systems	0.8655	0.676	0.676	0.676	0.676	0.8771	0.8771	0.9052	0.9052
SQL	Data management	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Dom.-specific progr. lang.	0.8655	0.6112	0	0	0.6112	0	0.8277	0.2616	0.8728
SQL	Databases	0.8655	0.4511	0	0	0.4511	0	0.6852	0.3417	0.7928
SQL	Information tech. manag.	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Project management	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Information retrieval	0.8655	0.3496	0	0	0.3496	0	0.5754	0.3924	0.742
SQL	Computer data	0.8655	0.1097	0	0	0.1097	0	0.225	0.5124	0.6221
SQL	Data	0.8655	0.1097	0	0	0.1097	0	0.225	0.5124	0.6221
SQL	Programming languages	0.8655	0.4908	0	0	0.4908	0	0.7237	0.3218	0.8126
SQL	Computer science	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Information technology	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Management	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Production and manufact.	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Product development	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Information science	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Computing	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Information	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Programming lang. topics	0.8655	0.451	0	0	0.451	0	0.6851	0.3417	0.7927
SQL	Computer languages	0.8655	0.4018	0	0	0.4018	0	0.6341	0.3663	0.7681
SQL	Computer programming	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Formal sciences	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Applied sciences	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Accountability	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Business	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Business economics	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Industry	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Product management	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Science studies	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Interdisciplinary fields	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Library and inf. science	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Digital technology	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Concepts	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Software engineering	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Notation	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Constructed languages	0.8655	0.3829	0	0	0.3829	0	0.6134	0.3758	0.7587
SQL	Formalism (deductive)	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Abstraction	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Scientific disciplines	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Academic disciplines	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Applied disciplines	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Social philosophy	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Corruption	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Core issues in ethics	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Society	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Economics	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Technology	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Branding	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Strategic management	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Science	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Libraries	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Technology by type	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Electronics	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Numbers	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Thought	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Systems engineering	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Conceptual systems	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Scientific modeling	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Communication	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Information systems	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Languages	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Theories of deduction	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Formalism	0.8655	0.9176	0	0	0.9176	0	1.0292	0.1084	1.026
SQL	Innovation	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Structure	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Creativity	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Problem solving	0.8655	0.7919	0	0	0.7919	0	0.9556	0.1713	0.9632
SQL	Academia	0.8655	0	0	0	0	0	0	0.5672	0.5672
SQL	Knowledge	0.8655	0	0	0	0	0	0	0.5672	0.5672

tenido de información, realizando una exploración básica y comparando con la categoría actual.

Como podemos observar en los datos obtenidos volvemos a tener el problema de la falta de valores para el IC de los *lcs*. Al estar basadas en la topología de la jerarquía, estas medidas tienen una alta dependencia de este factor. Si modificamos la topología original del grafo los valores se desvirtúan y por tanto dejan de tener utilidad. Aplicar las medidas sobre la estructura original tampoco tiene sentido, debido a la multitud de ciclos existentes.

En definitiva la conclusión es que estas medidas no se adaptan bien a su empleo con el grafo de categorías Wikipedia, al menos tal y como están implementadas actualmente. Aunque la idea en la que se basan es sólida, es de difícil aplicación a grafos de estas características.

5.2.6. Métodos basados en Recuperación de Información

En esta sección nos vamos a centrar en establecer una comparación entre categorías en base al contenido textual de éstas, para ello emplearemos técnicas de reconocido valor en recuperación de información. La idea básica es la de obtener los textos asociados a una categoría, es decir el texto de las páginas asociadas a ella y mediante una medida obtener el grado de similitud de las categorías en base al criterio impuesto por la medida.

Lo primero que debemos tener en cuenta es la fuente del texto que vamos a procesar, como ya hemos visto anteriormente hay dos fuentes principales de las que podemos obtener texto en una página de Wikipedia. Recordemos que definíamos una página como $p = (t, c, L, LC)$, donde t representa el título de la página y c representa su contenido, esto es, el texto de la página.

Así pues vamos a plantear dos posibilidades distintas:

- Procesar el texto contenido en los títulos de las páginas.
- Procesar el texto del contenido de las páginas.

Los criterios de comparación de nuevo van a venir determinados por los tres métodos de comparación que presentamos en la Sección 5.2.3, recordemos:

- *Comparación con la categoría inicial.* En esta variante compararemos el contenido textual de la categoría a evaluar (proveniendo el texto de una de la dos localizaciones vistas anteriormente) con la categoría inicial a partir de la cual se comenzó la exploración del grafo.
- *Comparación con la categoría actual.* En este caso compararemos los contenidos textuales de una categoría con el contenido de sus categorías padre seleccionado aquellas más similares y descartando aquellas no relacionadas.
- *Comparación con la categoría acumulada.* Esta última posibilidad propone la creación de una categoría de referencia compuesta a partir de la unión de todos los textos correspondientes a cada una de las categorías previamente seleccionadas. De este modo cada nueva categoría a evaluar se compara con todas las aceptadas hasta el momento.

Tabla 5.8: Valores de similitud usando medidas de contenido de información, comparando con la categoría actual

C_1	C_2	IC(C_1)	IC(C_2)	Resnik	R.IC(C_2)	Lin	L.IC(C_2)	J-C	JC.IC(C_2)
SQL	SQL	0.8655	0.8655	0.8655	0.8655	1	1	1	1
SQL	Database manag. syst.	0.8655	0.676	0.676	0.676	0.8771	0.8771	0.9052	0.9052
SQL	Query languages	0.8655	0.8201	0	0.8201	0	0.973	0	0.9773
Database manag. syst.	Databases	0.676	0.4511	0	0.4511	0	0.8005	0	0.8875
Query languages	Data management	0.8201	0	0	0	0	0	0.59	0.59
Query languages	Dom.-specif. progr. lang.	0.8201	0.6112	0	0.6112	0	0.854	0	0.8955
Databases	Computer data	0.4511	0.1097	0	0.1097	0	0.3913	0	0.8293
Databases	Computer science	0.4511	0	0	0	0	0	0	0.7744
Databases	Data management	0.4511	0	0	0	0	0	0	0.7744
Data management	Information retrieval	0	0.3496	0	0.3496	0	0	0	1.1748
Data management	Data	0	0.1097	0	0.1097	0	0	0	1.0548
Data management	Project management	0	0	0	0	0	0	0	0
Data management	Computer data	0	0.1097	0	0.1097	0	0	0	1.0549
Data management	Information tech. manag.	0	0	0	0	0	0	0	0
Dom.-specif. progr. lang.	Programming languages	0.6112	0.4908	0.4908	0.4908	0.8908	0.8908	0.9398	0.9398
Computer data	Computing	0.1097	0	0	0	0	0	0	0
Computer data	Data	0.1097	0.1097	0	0.1097	0	0.9999	0	1
Computer science	Formal sciences	0	0	0	0	0	0	0	0
Computer science	Applied sciences	0	0	0	0	0	0	0	0
Computer science	Computing	0	0	0	0	0	0	0	0
Information retrieval	Information science	0.3496	0	0	0	0	0	0	0
Data	Information	0.1097	0	0	0	0	0	0.9452	0.9452
Project management	Prod. and manufacturing	0	0	0	0	0	0	0	0
Project management	Management	0	0	0	0	0	0	0	0
Project management	Product development	0	0	0	0	0	0	0	0
Information tech. manag.	Project management	0	0	0	0	0	0	0	0
Information tech. manag.	Information technology	0	0	0	0	0	0	0	0
Programming languages	Programming lang. topics	0.4908	0.451	0	0.451	0	0.9577	0	0.9801
Programming languages	Computer programming	0.4908	0	0	0	0	0	0	0
Programming languages	Computer science	0.4908	0	0	0	0	0	0	0.7546
Programming languages	Computer languages	0.4908	0.4018	0	0.4018	0	0.9003	0	0.9555
Computing	Digital technology	0	0	0	0	0	0	0	0
Computing	Information technology	0	0	0	0	0	0	0	0
Formal sciences	Formalism (deductive)	0	0	0	0	0	0	0	0
Formal sciences	Interdisciplinary fields	0	0	0	0	0	0	0	0
Formal sciences	Academic disciplines	0	0	0	0	0	0	0	0
Formal sciences	Abstraction	0	0	0	0	0	0	0	0
Formal sciences	Scientific disciplines	0	0	0	0	0	0	0	0
Applied sciences	Applied disciplines	0	0	0	0	0	0	0	0
Applied sciences	Scientific disciplines	0	0	0	0	0	0	0	0
Information science	Information	0	0	0	0	0	0	0	0
Information science	Science studies	0	0	0	0	0	0	0	0
Information science	Library and inf. science	0	0	0	0	0	0	0	0
Information science	Interdisciplinary fields	0	0	0	0	0	0	0	0
Information	Concepts	0	0	0	0	0	0	0	0
Prod. and manufacturing	Industry	0	0	0	0	0	0	0	0
Management	Business	0	0	0	0	0	0	0	0
Management	Accountability	0	0	0	0	0	0	0	1
Management	Business economics	0	0	0	0	0	0	0	1
Management	Applied sciences	0	0	0	0	0	0	0	0
Product development	Prod. and manufacturing	0	0	0	0	0	0	0	0
Product development	Product management	0	0	0	0	0	0	0	0
Programming lang. topics	Software engineering	0.451	0	0	0	0	0	0	0
Programming lang. topics	Computer languages	0.451	0.4018	0	0.4018	0	0.9423	0	0.9754
Computer programming	Software engineering	0	0	0	0	0	0	0	0
Computer programming	Computing	0	0	0	0	0	0	0	0
Computer languages	Constructed languages	0.4018	0.3829	0	0.3829	0	0.9758	0	0.9905
Computer languages	Notation	0.4018	0	0	0	0	0	0	0.7991
Computer languages	Computing	0.4018	0	0	0	0	0	0	0
Digital technology	Technology by type	0	0	0	0	0	0	0	0
Digital technology	Electronics	0	0	0	0	0	0	0	0
Digital technology	Numbers	0	0	0	0	0	0	0	0
Formalism (deductive)	Theories of deduction	0	0	0	0	0	0	0	0
Formalism (deductive)	Formalism	0	0.9176	0	0.9176	0	0	0	0
Interdisciplinary fields	Science	0	0	0	0	0	0	1	1
Interdisciplinary fields	Academic disciplines	0	0	0	0	0	0	0	0
Academic disciplines	Academia	0	0	0	0	0	0	0	0
Abstraction	Structure	0	0	0	0	0	0	0	0
Abstraction	Problem solving	0	0.7919	0	0.7919	0	0	0	1.396
Abstraction	Innovation	0	0	0	0	0	0	0	1
Abstraction	Thought	0	0	0	0	0	0	0	1
Abstraction	Creativity	0	0	0	0	0	0	0	1
Scientific disciplines	Science	0	0	0	0	0	0	0	1
Applied disciplines	Knowledge	0	0	0	0	0	0	0	0
Applied disciplines	Academic disciplines	0	0	0	0	0	0	0	0
Science studies	Science	0	0	0	0	0	0	0	0
Science studies	Interdisciplinary fields	0	0	0	0	0	0	0	0

Ya tenemos el origen del texto a emplear, y la política de exploración y comparación de categorías. Sólo nos queda definir la representación de los textos y la medida con la que se va a evaluar la similitud entre categorías.

En recuperación de información una de las representaciones más utilizadas para tratar con textos es el modelo de espacio vectorial (*vector space model*). En este modelo se representa cada documento o texto como un vector de términos. Para comparar dos documentos, se aplica la medida del coseno sobre los vectores que representan a dichos documentos.

En nuestro caso vamos a representar cada categoría por un vector de términos y utilizaremos el modelo de espacio vectorial como en [Manning et al., 2008].

Nuestra representación definirá cada categoría por un vector de términos. Esos términos se recogerán según el caso, de los títulos o del contenido de las páginas clasificadas en la categoría correspondiente. Una vez tenemos ambos vectores se aplica la medida de similitud del coseno [Zobel and Moffat, 1998] tal como se formula a continuación:

$$sim_{cos}(c_1, c_2) = \frac{\vec{V}(c_1) \cdot \vec{V}(c_2)}{|\vec{V}(c_1)||\vec{V}(c_2)|} \tag{5.18}$$

donde el numerador representa el producto escalar de los vectores $\vec{V}(c_1)$ y $\vec{V}(c_2)$ y el denominador es el producto de sus distancias euclídeas
El producto escalar de dos vectores $\vec{x} \cdot \vec{y}$ se define como $\sum_{i=1}^{M} x_i y_i$.
Dado $\vec{V}(c)$ el vector para los textos de la categoría c, con M componentes $\vec{V}_1(c), \ldots, \vec{V}_M(c)$.
La distancia euclídea de c se como define como $\sqrt{\sum_{i=1}^{M} \vec{V}_i^2(c)}$.
El efecto que tiene el denominador de la Fórmula 5.18 es el de normalizar la longitud de los vectores $\vec{V}(c_1)$ y $\vec{V}(c_2)$ a vectores unidad $\vec{v}(c_1) = \vec{V}(c_1)/|\vec{V}(c_1)|$ y $\vec{v}(c_2) = \vec{V}(c_2)/|\vec{V}(c_2)|$.

Lo siguiente que debemos tener en cuenta es como se calculan los pesos de los vectores de términos. En recuperación de información se suele emplear habitualmente el esquema de pesos *tf-idf* (*term frequency-inverse term frequency*), para la búsqueda de documentos en una colección. Este esquema de pesos es uno de lo más extendidos y utilizados en la literatura sobre recuperación de información.

Se define *tf-idf* como el producto de la frecuencia de un término, por la frecuencia inversa del documento (en nuestro caso categoría):

$$\text{tf-idf}_{t,c} = \text{tf}_{t,c} \times \text{idf}_t. \tag{5.19}$$

La frecuencia de un término $\text{tf}_{t,c}$ consiste en el número de veces que aparece el término t en el texto de la categoría c. Para evitar dar mayor importancia a términos que aparecen muchas veces por estar en textos muy largos se normaliza la frecuencia por la longitud del texto y obtenemos:

$$\text{tf}_{t,c} = \frac{n_{t,c}}{\sum_k n_{k,c}}, \tag{5.20}$$

donde el numerador representa el número de veces que aparece el término t en el texto de la categoría c, y el denominador es el número de términos en el texto de la categoría c. De

este modo se normaliza la frecuencia de los términos por la longitud de los textos. Este valor determina cómo de importante es el término en el contexto del documento.

Otro modo de calcular la frecuencia, también normalizando, pero manteniendo los valores de frecuencia en el rango $[0, 1]$ es mediante una modificación de $\mathrm{tf}_{t,c}$ formulada de la siguiente manera:

$$\mathrm{wf}_{t,c} = \frac{n_{t,c}}{max_l\ n_{l,c}}, \tag{5.21}$$

donde el numerador representa la frecuencia, o número de veces que aparece el término t en el texto de la categoría c, y el denominador es la frecuencia máxima de un término en el documento.

Con respecto a idf_t, la frecuencia de documento inversa se define como:

$$\mathrm{idf}_t = \log \frac{N}{1 + \mathrm{df}_t}. \tag{5.22}$$

donde N es el número de categorías para las que estamos calculando los pesos de los vectores, y df_t es la frecuencia de documento del término t, que indica el número de documentos de la colección en los que aparece el término. Al valor de df_t se le suma 1, para evitar división por cero en los casos en que el término no aparece en los textos. Cabe destacar que el logaritmo empleado es en base 10, pero realmente la base del logaritmo no es un factor influyente en el resultado.

El valor de idf_t representa la importancia de un término dentro de la colección de textos, en nuestro caso, la relevancia de un término para el conjunto de categorías utilizadas en el cálculo de pesos.

El esquema de pesos *tf-idf* está especialmente indicado para la obtención de documentos en base a consultas textuales. Sin embargo en nuestro caso la aplicación estaría mas encaminada a la comparación de documentos, ya que no vamos a realizar consultas como tales, sino que vamos a comparar categorías. Para poder realizar esta comparación de forma efectiva sería necesario calcular los vectores con los pesos *tf-idf* para todas las categorías existentes. Esto implicaría un alto coste de procesamiento, por no mencionar la necesidad de recalcular los valores conforme se incluyesen nuevas páginas en las categorías.

Dado que este proceso es a todas luces muy costoso, nosotros vamos a emplear un esquema de pesos que nos permita comparar dos categorías sin necesidad de procesar todo el contenido textual del que se dispone. Para ello utilizaremos únicamente la frecuencia $\mathrm{wf}_{t,c}$ de los términos para construir los vectores de pesos.

Una vez visto el esquema que se va a emplear para el cálculo de la similitud entre categorías vamos a ver los resultados obtenidos con cada una de las variantes.

Selección de Categorías empleando Medidas basadas en Recuperación de Información

En este apartado vamos a realizar algunos experimentos con el objetivo de obtener valores de similitud entre categorías, para poder determinar si el método empleado es adecuado.

Como en los casos anteriores comenzaremos con una exploración del grafo siguiendo el método de exploración básico y seleccionando las categorías adecuadas en base al valor que obtengamos al compararlas con la referencia que hayamos elegido.

Comenzaremos realizando la comparación con la categoría inicial, y teniendo en cuenta como textos a procesar, por una parte los títulos de las páginas de la categoría correspondiente, y por otra los contenidos de los textos de las páginas de Wikipedia. Adicionalmente obtendremos datos de similitud obtenidos al añadir un proceso de lematización al cálculo de resultados.

Atendiendo a estos parámetros obtenemos la Tabla 5.9.

Como se puede observar por los datos obtenidos, la evaluación del contenido completo de la página, que incluye también el título, da mejores resultados que el procesamiento de únicamente los títulos. Este fenómeno se debe a que en ocasiones si bien los títulos son completamente descriptivos, por ejemplo, el nombre de un sistema gestor de bases de datos, esta información no posee semántica adicional. Sin embargo, en el contenido de la página correspondiente se realiza una descripción de las funciones y componentes de dicho sistema, lo que permite una mejor caracterización semántica del título en sí mismo.

Con respecto al uso de la lematización en el procesamiento de los datos, atendiendo a los valores obtenidos, su uso no representa un incremento significativo de los resultados, la diferencia de los resultados en media es de 0.026. Sin embargo, el proceso de lematización introduce un incremento de procesamiento, así como cierta ambigüedad al caracterizar como equivalentes términos diferentes con idéntica raíz.

Atendiendo a los resultados obtenidos parece que utilizar el contenido de las páginas sin lematizar, va a ser la mejor estrategia a seguir, veamos si esta primera apreciación se confirma en los siguientes métodos.

Comprobemos ahora los resultados obtenidos al cambiar de estrategia y realizar la comparación entre categorías relacionadas directamente (padre-hija). Los resultados se muestran en la tabla 5.10.

En esta ocasión vemos como el comportamiento de los datos es similar al del caso anterior pero con algunas excepciones. Una diferencia con respecto al caso anterior es que el procesamiento de títulos sin lematizar mejora en más casos, aunque en media sigue ofreciendo mejores valores la versión lematizada. Con respecto a la selección del origen del texto sigue siendo mejor emplear el texto completo de la página frente a usar únicamente el título. Con respecto a la diferencia entre usar lematización o no con los textos de las páginas, al igual que en el caso anterior la diferencia no es particularmente relevante aunque la diferencia en media aumenta ligeramente hasta obtener un valor de 0.029.

Así pues, seguimos considerando que la mejor opción es emplear el texto completo de las páginas sin lematizar, para obtener unos valores de similitud apropiados.

Finalmente vamos a probar con la exploración acumulativa, donde vamos a crear una categoría abstracta formada por los textos de todas las categorías que han sido aceptadas previamente. La Tabla 5.11 muestra los resultados obtenidos.

Los resultados obtenidos en este caso nos indican de nuevo que es mejor emplear el texto completo de las páginas para obtener valores de relación más cercanos a los que asignaría un experto. Con respecto a la conveniencia de usar lematización, en este caso la media de la diferencia baja hasta 0.015, por lo que al igual que antes vamos a decantarnos

Tabla 5.9: Valores de similitud utilizando la medida del coseno y comparando con la categoría inicial

C_1	C_2	Título	Título Stem	Contenido	Contenido Stem
SQL	SQL	1	1	1	1
SQL	Query languages	0.371164	0.367437	0.827852	0.827352
SQL	Database management systems	0.304376	0.328664	0.677177	0.707665
SQL	Data management	0.252432	0.261008	0.48218	0.524522
SQL	Domain-specific progr. lang.	0.143595	0.143456	0.391762	0.438828
SQL	Databases	0.259525	0.285854	0.607517	0.637537
SQL	Information tech. management	0.080488	0.087887	0.331229	0.375797
SQL	Project management	0.045397	0.054935	0.220149	0.267638
SQL	Information retrieval	0.026042	0.034863	0.28368	0.330046
SQL	Computer data	0.15534	0.146782	0.37283	0.407454
SQL	Data	0.183383	0.18583	0.319919	0.331589
SQL	Programming languages	0.069589	0.069135	0.230309	0.282883
SQL	Computer science	0.015965	0.016131	0.235931	0.259773
SQL	Information technology	0	0	0.174878	0.21601
SQL	Management	0.059468	0.064298	0.262228	0.307915
SQL	Production and manufacturing	0.03604	0.048943	0.302411	0.338459
SQL	Product development	0.007698	0.012747	0.245017	0.289058
SQL	Information science	0.028382	0.032299	0.284907	0.337122
SQL	Computing	0.024714	0.021124	0.346479	0.357275
SQL	Information	0.008995	0.01619	0.192546	0.239355
SQL	Programming language topics	0.112747	0.134682	0.355021	0.401475
SQL	Computer languages	0.143208	0.150062	0.444754	0.488037
SQL	Computer programming	0.053736	0.056985	0.418974	0.444142
SQL	Formal sciences	0.051333	0.050225	0.312836	0.343224
SQL	Applied sciences	0.015462	0.019024	0.266593	0.294317
SQL	Accountability	0	0	0.127105	0.149626
SQL	Business	0.045087	0.046779	0.248313	0.285035
SQL	Business economics	0.012329	0.017822	0.242244	0.268359
SQL	Industry	0.008219	0.008447	0.25605	0.271739
SQL	Product management	0.034662	0.03948	0.22693	0.258233
SQL	Science studies	0	0.004945	0.180037	0.21213
SQL	Interdisciplinary fields	0.003255	0.00851	0.213505	0.241849
SQL	Library and information science	0.042771	0.052655	0.24931	0.283882
SQL	Digital technology	0.016294	0.018211	0.241216	0.285712
SQL	Concepts	0.010849	0.014472	0.204411	0.223563
SQL	Software engineering	0.007842	0.016751	0.251934	0.309096
SQL	Notation	0.01408	0.013099	0.294029	0.331961
SQL	Constructed languages	0.119416	0.130216	0.235347	0.26315
SQL	Formalism (deductive)	0	0	0.124249	0.159996
SQL	Abstraction	0.006657	0.009694	0.21549	0.247307
SQL	Scientific disciplines	-1	-1	-1	-1
SQL	Academic disciplines	0	0	0.18708	0.208814
SQL	Applied disciplines	-1	-1	-1	-1
SQL	Social philosophy	0.005142	0.00858	0.238216	0.261346
SQL	Corruption	0	0.006994	0.215465	0.238504
SQL	Core issues in ethics	0.004476	0.004279	0.269043	0.287773
SQL	Society	0	0	0.165758	0.168309
SQL	Economics	0.004781	0.008467	0.226053	0.24357
SQL	Technology	0.02138	0.022321	0.279444	0.30875
SQL	Branding	0.005629	0.006575	0.199041	0.205378
SQL	Strategic management	0.030945	0.03938	0.233189	0.270035
SQL	Science	0	0	0.222124	0.245381
SQL	Libraries	0.001766	0.009169	0.195172	0.199713
SQL	Technology by type	0	0	0.189636	0.224231
SQL	Electronics	0.006433	0.011471	0.193082	0.215781
SQL	Numbers	0.003836	0.003605	0.134703	0.149089
SQL	Thought	0.005443	0.019116	0.213327	0.261507
SQL	Systems engineering	0.034933	0.048826	0.278478	0.327082
SQL	Conceptual systems	0.012797	0.010951	0.194948	0.231595
SQL	Scientific modeling	0.019018	0.039597	0.28782	0.284886
SQL	Communication	0.018383	0.023538	0.243078	0.273658
SQL	Information systems	0.02126	0.022395	0.300715	0.341412
SQL	Languages	0.145732	0.150193	0.209638	0.230338
SQL	Theories of deduction	0.004215	0.007563	0.174625	0.204312
SQL	Formalism	0	0	0.083942	0.100596
SQL	Innovation	0.007858	0.011344	0.204419	0.23412
SQL	Structure	0.036108	0.032335	0.216176	0.23708
SQL	Creativity	0.003042	0.01039	0.221784	0.23897
SQL	Problem solving	0.001466	0.03041	0.235566	0.274921
SQL	Academia	0.009864	0.015264	0.204015	0.224098
SQL	Knowledge	0.003265	0.006483	0.19322	0.232299

Tabla 5.10: Valores de similitud utilizando la medida del coseno y comparando con la categoría actual

C_1	C_2	Título	Título Stem	Contenido	Cont. Stem
SQL	SQL	1	1	1	1
SQL	Database management systems	0.304376	0.328664	0.677177	0.707665
SQL	Query languages	0.371164	0.367437	0.827852	0.827352
Database management systems	Databases	0.892666	0.899408	0.769328	0.797359
Query languages	Data management	0.132704	0.138707	0.485707	0.519533
Query languages	Domain-specific progr. lang.	0.596813	0.615075	0.584702	0.640474
Databases	Computer data	0.277491	0.247963	0.629324	0.613666
Databases	Computer science	0.028494	0.042263	0.403903	0.417417
Databases	Data management	0.522408	0.514267	0.821896	0.829481
Data management	Information retrieval	0.088216	0.100013	0.43692	0.465145
Data management	Data	0.803786	0.796463	0.797424	0.755998
Data management	Project management	0.109177	0.127823	0.327099	0.387746
Data management	Computer data	0.782773	0.747379	0.877312	0.860367
Data management	Information tech. management	0.293526	0.322526	0.62889	0.662024
Domain-specific progr. lang.	Programming languages	0.497085	0.496621	0.560899	0.642089
Computer data	Computing	0.345238	0.455776	0.505093	0.540648
Computer data	Data	0.697769	0.65835	0.816976	0.78809
Computer science	Formal sciences	0.313014	0.380175	0.58991	0.652623
Computer science	Applied sciences	0.380737	0.3669	0.395112	0.388474
Computer science	Computing	0.331721	0.612425	0.651178	0.769369
Information retrieval	Information science	0.380578	0.398468	0.667745	0.668404
Data	Information	0.116781	0.115412	0.371772	0.426646
Project management	Production and manufacturing	0.339065	0.375296	0.634544	0.660427
Project management	Management	0.633094	0.658671	0.710356	0.740569
Project management	Product development	0.099551	0.110693	0.478313	0.515428
Information tech. management	Project management	0.56776	0.585897	0.615111	0.658131
Information tech. management	Information technology	0.283871	0.2936	0.593609	0.615588
Programming languages	Programming language topics	0.46291	0.495351	0.557495	0.652799
Programming languages	Computer programming	0.339422	0.353253	0.549872	0.627985
Programming languages	Computer science	0.032774	0.028571	0.282119	0.316411
Programming languages	Computer languages	0.374166	0.377127	0.529434	0.603907
Computing	Digital technology	0.022273	0.017688	0.502949	0.520097
Computing	Information technology	0.00607	0.006812	0.402967	0.425397
Formal sciences	Formalism (deductive)	0.012982	0.255697	0.366737	0.45823
Formal sciences	Interdisciplinary fields	0.126091	0.149776	0.601835	0.63658
Formal sciences	Academic disciplines	0.062257	0.058909	0.551427	0.554904
Formal sciences	Abstraction	0.047749	0.073531	0.625494	0.658125
Formal sciences	Scientific disciplines	-1	-1	-1	-1
Applied sciences	Applied disciplines	-1	-1	-1	-1
Applied sciences	Scientific disciplines	-1	-1	-1	-1
Information science	Information	0.733911	0.756794	0.843111	0.848336
Information science	Science studies	0.229521	0.226224	0.507931	0.55939
Information science	Library and information science	0.37243	0.357558	0.718539	0.70134
Information science	Interdisciplinary fields	0.119425	0.149779	0.524447	0.581429
Information	Concepts	0.034544	0.040109	0.328778	0.372178
Production and manufacturing	Industry	0.334258	0.323528	0.821572	0.831221
Management	Business	0.515309	0.51874	0.807687	0.837719
Management	Accountability	0.00789	0.017303	0.309289	0.34578
Management	Business economics	0.119122	0.138799	0.639037	0.651661
Management	Applied sciences	0.059859	0.06473	0.480441	0.519124
Product development	Production and manufacturing	0.134512	0.168837	0.63408	0.690635
Product development	Product management	0.469521	0.479216	0.733263	0.762032
Programming language topics	Software engineering	0.068126	0.073915	0.343784	0.398164
Programming language topics	Computer languages	0.628293	0.718516	0.779052	0.831248
Computer programming	Software engineering	0.146886	0.138194	0.478342	0.502169
Computer programming	Computing	0.275743	0.372619	0.664065	0.683313
Computer languages	Constructed languages	0.623729	0.677029	0.615072	0.668688
Computer languages	Notation	0.075703	0.071453	0.61608	0.695488
Computer languages	Computing	0.099662	0.138274	0.538598	0.562938
Digital technology	Technology by type	0.055225	0.058076	0.441583	0.501271
Digital technology	Electronics	0.006958	0.099901	0.34649	0.413963
Digital technology	Numbers	0	0	0.155995	0.178852
Formalism (deductive)	Theories of deduction	0.020253	0.018288	0.345585	0.430728
Formalism (deductive)	Formalism	0.736075	0.736075	0.282171	0.332505
Interdisciplinary fields	Science	0.141954	0.16724	0.600879	0.649294
Interdisciplinary fields	Academic disciplines	0.589675	0.584759	0.688798	0.722291
Academic disciplines	Academia	0.254348	0.246497	0.531081	0.545426
Abstraction	Structure	0.127631	0.131168	0.396312	0.41262
Abstraction	Problem solving	0.062192	0.074015	0.522403	0.564591
Abstraction	Innovation	0.004629	0.004383	0.317201	0.327542
Abstraction	Thought	0.096205	0.162478	0.647879	0.694428
Abstraction	Creativity	0.043011	0.058705	0.473352	0.479047
Scientific disciplines	Science	-1	-1	-1	-1
Applied disciplines	Knowledge	-1	-1	-1	-1
Applied disciplines	Academic disciplines	-1	-1	-1	-1
Science studies	Science	0.398527	0.390513	0.798803	0.825923
Science studies	Interdisciplinary fields	0.331764	0.357568	0.685087	0.732098

Tabla 5.11: Valores de similitud utilizando la medida del coseno y comparando con la categoría acumulada. Umbral $t = 0$

C_1	C_2	Título	Título Stem	Contenido	Contenido Stem
SQL	SQL	1	1	1	1
Acumulado	Query languages	0.371164	0.367437	0.827852	0.827352
Acumulado	Database management systems	0.327942	0.351925	0.681302	0.706528
Acumulado	Data management	0.437959	0.446218	0.644862	0.677359
Acumulado	Domain-specific progr. lang.	0.189186	0.199383	0.491127	0.519241
Acumulado	Databases	0.691516	0.69215	0.874227	0.88207
Acumulado	Information technology manag.	0.279342	0.295601	0.57816	0.609254
Acumulado	Project management	0.289615	0.310336	0.446148	0.4979
Acumulado	Information retrieval	0.137922	0.15538	0.507571	0.533955
Acumulado	Computer data	0.445687	0.418896	0.70128	0.701084
Acumulado	Data	0.531519	0.51246	0.643685	0.621554
Acumulado	Programming languages	0.099156	0.104455	0.334035	0.372
Acumulado	Computer science	0.095512	0.125232	0.47663	0.496003
Acumulado	Information technology	0.163572	0.162198	0.473734	0.51359
Acumulado	Management	0.506998	0.521892	0.656218	0.698284
Acumulado	Production and manufacturing	0.397228	0.44749	0.739812	0.76375
Acumulado	Product development	0.078611	0.091338	0.559222	0.605722
Acumulado	Information science	0.274618	0.288351	0.680288	0.718743
Acumulado	Computing	0.130903	0.162256	0.661451	0.639176
Acumulado	Information	0.339832	0.384237	0.596389	0.643617
Acumulado	Programming language topics	0.160787	0.201369	0.488255	0.50473
Acumulado	Computer languages	0.238024	0.310756	0.564037	0.601707
Acumulado	Computer programming	0.247213	0.276049	0.641924	0.642651
Acumulado	Formal sciences	0.23799	0.281699	0.686153	0.700282
Acumulado	Applied sciences	0.156195	0.169609	0.570522	0.597699
Acumulado	Accountability	0.068976	0.072004	0.377388	0.387633
Acumulado	Business	0.354924	0.354976	0.671301	0.703344
Acumulado	Business economics	0.126084	0.155757	0.621593	0.636309
Acumulado	Industry	0.120654	0.125547	0.65731	0.658927
Acumulado	Product management	0.381494	0.411413	0.608498	0.641803
Acumulado	Science studies	0.122766	0.134892	0.47168	0.502151
Acumulado	Interdisciplinary fields	0.125209	0.160494	0.557145	0.578208
Acumulado	Library and information science	0.26901	0.277139	0.59249	0.593142
Acumulado	Digital technology	0.084833	0.090456	0.538354	0.592265
Acumulado	Concepts	0.058823	0.066023	0.505413	0.514445
Acumulado	Software engineering	0.164487	0.199809	0.540104	0.607935
Acumulado	Notation	0.050904	0.060588	0.481741	0.501981
Acumulado	Constructed languages	0.167363	0.18824	0.410563	0.403475
Acumulado	Formalism (deductive)	0.003554	0.013602	0.289182	0.33764
Acumulado	Abstraction	0.042029	0.05721	0.512159	0.538632
Acumulado	Scientific disciplines	-1	-1	-1	-1
Acumulado	Academic disciplines	0.158491	0.161503	0.561643	0.563688
Acumulado	Applied disciplines	-1	-1	-1	-1
Acumulado	Social philosophy	0.116056	0.129719	0.628043	0.638044
Acumulado	Corruption	0.039096	0.05427	0.526848	0.540871
Acumulado	Core issues in ethics	0.064109	0.083801	0.695715	0.696117
Acumulado	Society	0.051697	0.048884	0.495446	0.463496
Acumulado	Economics	0.13573	0.144158	0.657753	0.661327
Acumulado	Technology	0.257459	0.263857	0.719346	0.727904
Acumulado	Branding	0.081884	0.077148	0.485751	0.473202
Acumulado	Strategic management	0.436191	0.421189	0.68576	0.708269
Acumulado	Science	0.155356	0.181164	0.6256	0.646649
Acumulado	Libraries	0.077927	0.105587	0.377802	0.367713
Acumulado	Technology by type	0.150487	0.169675	0.558182	0.595147
Acumulado	Electronics	0.135963	0.143378	0.452894	0.462237
Acumulado	Numbers	0.018313	0.022222	0.281001	0.270355
Acumulado	Thought	0.088857	0.148526	0.632018	0.689373
Acumulado	Systems engineering	0.360133	0.391528	0.685933	0.708911
Acumulado	Conceptual systems	0.234271	0.280634	0.555828	0.585224
Acumulado	Scientific modeling	0.256965	0.288032	0.707746	0.649353
Acumulado	Communication	0.214659	0.248165	0.672867	0.686716
Acumulado	Information systems	0.493789	0.513736	0.745956	0.761261
Acumulado	Languages	0.20565	0.209586	0.337866	0.322582
Acumulado	Theories of deduction	0.070368	0.087992	0.464519	0.484816
Acumulado	Formalism	0.024619	0.034903	0.264533	0.286538
Acumulado	Innovation	0.189453	0.216793	0.597244	0.618486
Acumulado	Structure	0.118094	0.110393	0.515412	0.515151
Acumulado	Creativity	0.10001	0.117684	0.632561	0.631829
Acumulado	Problem solving	0.130528	0.174538	0.670811	0.707744
Acumulado	Academia	0.157911	0.170979	0.581411	0.589117
Acumulado	Knowledge	0.095034	0.109096	0.603933	0.652238

por no realizar lematización, con el objetivo de preservar el máximo de términos.

En este método el conjunto de categorías aceptadas conforma un contexto para la aceptación de la siguiente categoría. La aceptación de una categoría viene dada por la superación de un umbral de similitud, en el caso anterior el umbral era 0.

En la Figura 5.27 presentamos los resultados de los tres métodos para establecer una comparación. Para cada categoría se muestra el grado de relación para su selección como parte del camino que define a *SQL*. Se ha optado por presentar los valores en forma de un diagrama de líneas, a pesar de que no son valores continuos, para poder representar el valor de similitud o aceptación para cada categoría y ver cómo evolucionan estos según se va explorando el grafo. Para ello, el orden del eje X es el que viene dado por la forma en que se recorre el grafo. Para la elaboración de la gráfica se han tomado las primeras 57 categorías visitadas. Este número es muy elevado y en ningún caso es deseable seleccionar tantas categorías para una única categoría inicial, sin embargo resulta útil ver la evolución de las medidas en cuanto se adentran en zonas altas del grafo.

Como se puede observar el método de comparación con la categoría inicial da unos valores bajos de relación en cuanto nos vamos alejando de dicha categoría, aunque este comportamiento es adecuado los valores son tan bajos que apenas suponen un indicativo de relación en los casos en que sí existe. Este método es útil si se desea realizar una búsqueda local, pero resulta poco eficiente si se desea obtener un contexto más global. Por otra parte, al usar la comparación con la categoría inmediatamente anterior se obtienen valores muy altos, puesto que lo que se está haciendo es una búsqueda local en la que se va actualizando el contexto de búsqueda. El inconveniente de este método es precisamente esa continua actualización, puesto que esto implica que en cada paso hay nuevas categorías que incluir, y se aceptará una gran cantidad de categorías. Por último, el método acumulativo representa un punto intermedio entre las dos aproximaciones anteriores, si bien el contexto también es cambiante como en el segundo caso, ahora la evaluación de una categoría dependerá de todas las anteriores, contextualizando la comparación de una forma más adecuada.

En los dos primeros métodos, la aplicación de un umbral nos permite descartar todas aquellas categorías que no lo superan. Sin embargo los valores de la comparación no van a variar por la aplicación de dicho umbral. Sin embargo, en el caso de la categoría acumulada, el contexto se va a crear en base a dicho umbral y por tanto los valores de comparación irán variando en base a éste.

Estudiaremos el comportamiento del tercer método variando el valor de umbral. En la Tabla 5.12 mostramos los valores obtenidos con la medida del coseno empleando el texto completo de las páginas y sin usar lematización. La categoría inicial es *SQL*.

En la tabla vemos como valores de umbral de 0.4 y menores producen unos resultados muy similares a los obtenidos para umbral 0.

Para facilitar la comprensión de los datos presentados, en la Figura 5.28 se muestra una gráfica en la que se muestran los valores de relación para la categoría acumulada con umbrales 0, 0.6 y 0.8, los valores para 0.2 y 04, no se muestran por ser prácticamente equivalentes a los obtenidos para el umbral 0.

La gráfica muestra como los valores para el umbral 0.6, ofrecen unos valores adecuados. Este umbral es algo elevado pero debemos de tener en cuenta que cuanto más relajemos ese

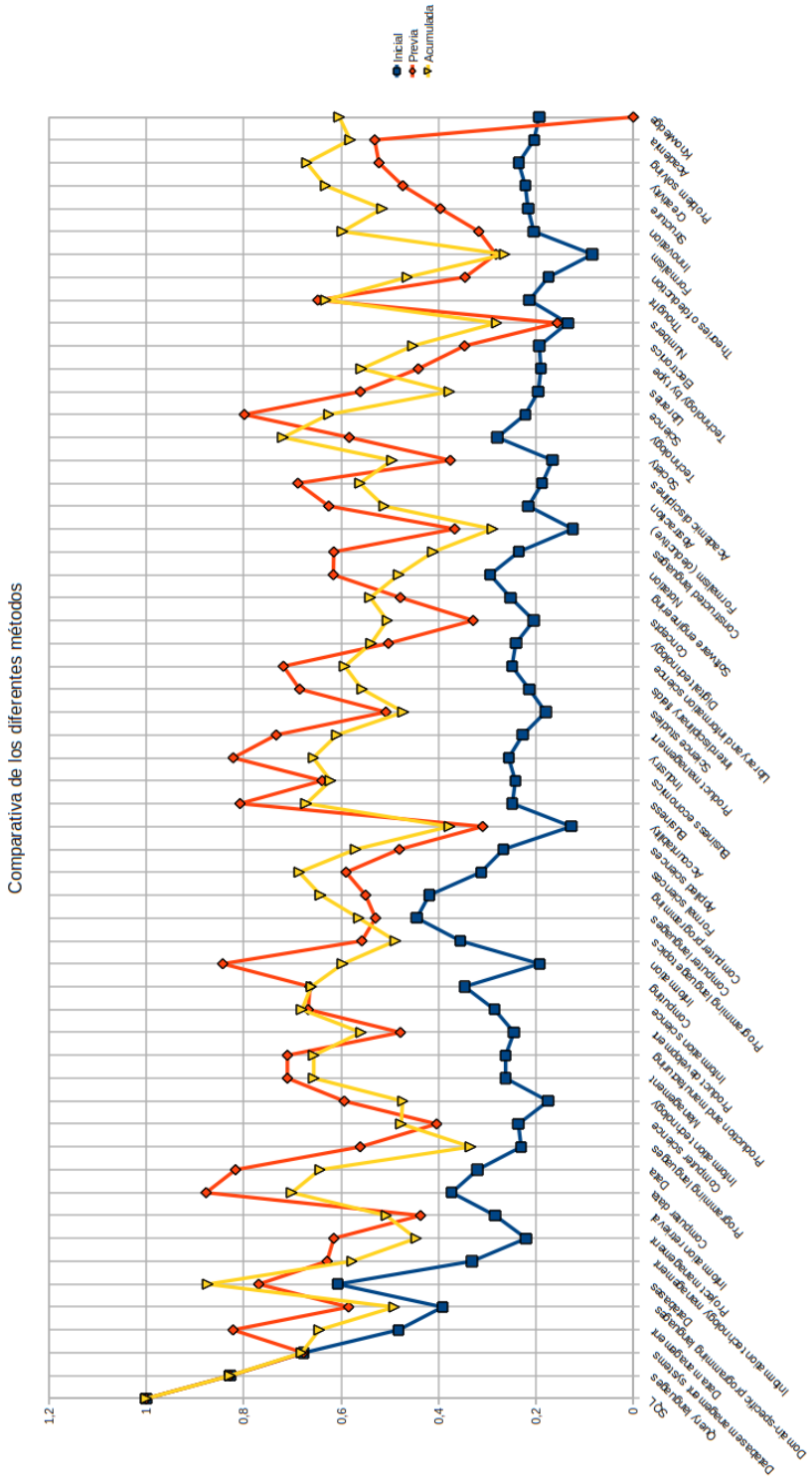


Figura 5.27: Valores obtenidos para las categorías para cada uno de los métodos

Tabla 5.12: Valores de similitud utilizando la medida del coseno y el contenido de las páginas sin lematizar

C1	C2	t=0.2	t=0.4	t=0.5	t=0.6	t=0.8
SQL	SQL	1	1	1	1	1
Acumulado	Query languages	0.827852	0.827852	0.827852	0.827852	0.827852
Acumulado	Database management systems	0.681302	0.681302	0.681302	0.681302	0.681302
Acumulado	Data management	0.644862	0.644862	0.644862	0.644862	0.505226
Acumulado	Domain-specific progr. lang.	0.491127	0.491127	0.491127	0.491127	0.492952
Acumulado	Databases	0.874227	0.874227	0.877316	0.877316	0.618161
Acumulado	Information technology management	0.57816	0.57816	0.573712	0.573712	0.348162
Acumulado	Project management	0.446148	0.446148	0.443339	0.337646	0.235174
Acumulado	Information retrieval	0.507571	0.507571	0.503546	0.45288	0.342017
Acumulado	Computer data	0.70128	0.70128	0.732226	0.730017	0.398294
Acumulado	Data	0.643685	0.643685	0.67328	0.676484	0.341691
Acumulado	Programming languages	0.334035	0.334035	0.317617	0.316505	0.297461
Acumulado	Computer science	0.47663	0.476293	0.464142	0.420397	0.274925
Acumulado	Information technology	0.473734	0.473855	0.46596	0.334445	0.199953
Acumulado	Management	0.656218	0.656345	0.571243	0.429472	0.279559
Acumulado	Production and manufacturing	0.739812	0.739798	0.70681	0.492438	0.328966
Acumulado	Product development	0.559222	0.559178	0.537218	0.364294	0.270105
Acumulado	Information science	0.680288	0.680311	0.666512	0.526929	0.327811
Acumulado	Computing	0.661451	0.66118	0.653485	0.564401	0.387709
Acumulado	Information	0.596389	0.596494	0.587402	0.394274	0.221543
Acumulado	Programming language topics	0.488255	0.487535	0.47668	0.474777	0.460882
Acumulado	Computer languages	0.564037	0.563459	0.520611	0.522679	0.553718
Acumulado	Computer programming	0.641924	0.641365	0.596103	0.570669	0.488518
Acumulado	Formal sciences	0.686153	0.686118	0.684331	0.594602	0.354772
Acumulado	Applied sciences	0.570522	0.57053	0.566903	0.439905	0.303144
Acumulado	Accountability	0.377388	0.377461	0.376322	0.24507	0.146593
Acumulado	Business	0.671301	0.670825	0.669128	0.39943	0.270366
Acumulado	Business economics	0.621593	0.621477	0.622345	0.395208	0.26741
Acumulado	Industry	0.65731	0.657109	0.656376	0.426892	0.282489
Acumulado	Product management	0.608498	0.608809	0.6066	0.376171	0.246094
Acumulado	Science studies	0.47168	0.471324	0.467862	0.31135	0.210949
Acumulado	Interdisciplinary fields	0.557145	0.5564	0.532747	0.345273	0.255409
Acumulado	Library and information science	0.59249	0.590447	0.584793	0.413647	0.28579
Acumulado	Digital technology	0.538354	0.538121	0.537951	0.413209	0.268123
Acumulado	Concepts	0.505413	0.503952	0.494814	0.338386	0.243458
Acumulado	Software engineering	0.540104	0.540528	0.525315	0.379556	0.27513
Acumulado	Notation	0.481741	0.481835	0.452884	0.368361	0.385808
Acumulado	Constructed languages	0.410563	0.410418	0.380186	0.305618	0.339114
Acumulado	Formalism (deductive)	0.289182	0.289268	0.276465	0.192975	0.143383
Acumulado	Abstraction	0.512159	0.51141	0.485561	0.34959	0.261521
Acumulado	Scientific disciplines	-1	-1	-1	-1	-1
Acumulado	Academic disciplines	0.561643	0.561176	0.545048	0.346249	0.211876
Acumulado	Applied disciplines	-1	-1	-1	-1	-1
Acumulado	Social philosophy	0.628043	0.626403	0.602468	0.38135	0.274228
Acumulado	Corruption	0.526848	0.524998	0.524563	0.369256	0.239994
Acumulado	Core issues in ethics	0.695715	0.694405	0.68571	0.429463	0.307867
Acumulado	Society	0.495446	0.494694	0.492781	0.271146	0.190729
Acumulado	Economics	0.657753	0.657126	0.659287	0.377689	0.253696
Acumulado	Technology	0.719346	0.71965	0.717005	0.476493	0.310554
Acumulado	Branding	0.485751	0.485703	0.489524	0.298298	0.225891
Acumulado	Strategic management	0.68576	0.686278	0.690635	0.3825	0.249799
Acumulado	Science	0.6256	0.625516	0.61336	0.397075	0.253671
Acumulado	Libraries	0.377802	0.37748	0.378249	0.250344	0.234054
Acumulado	Technology by type	0.558182	0.559293	0.563224	0.326102	0.206964
Acumulado	Electronics	0.452894	0.452229	0.448985	0.30129	0.218783
Acumulado	Numbers	0.281001	0.280339	0.275172	0.22455	0.148098
Acumulado	Thought	0.632018	0.63196	0.620816	0.360531	0.248318
Acumulado	Systems engineering	0.685933	0.688806	0.684854	0.471572	0.313051
Acumulado	Conceptual systems	0.555828	0.55628	0.553702	0.352974	0.221465
Acumulado	Scientific modeling	0.707746	0.7089	0.705933	0.516702	0.332176
Acumulado	Communication	0.672867	0.672016	0.666364	0.40488	0.285793
Acumulado	Information systems	0.745956	0.748236	0.758764	0.574107	0.328681
Acumulado	Languages	0.337866	0.336823	0.300936	0.23629	0.323633
Acumulado	Theories of deduction	0.464519	0.461795	0.446315	0.272026	0.216033
Acumulado	Formalism	0.264533	0.263744	0.258391	0.142332	0.096116
Acumulado	Innovation	0.597244	0.599286	0.601217	0.332931	0.225024
Acumulado	Structure	0.515412	0.515123	0.512773	0.387891	0.248661
Acumulado	Creativity	0.632561	0.63123	0.626502	0.366636	0.253686
Acumulado	Problem solving	0.670811	0.670663	0.667011	0.395471	0.268009
Acumulado	Academia	0.581411	0.578411	0.576784	0.32258	0.232297
Acumulado	Knowledge	0.603933	0.603311	0.600913	0.337298	0.222631

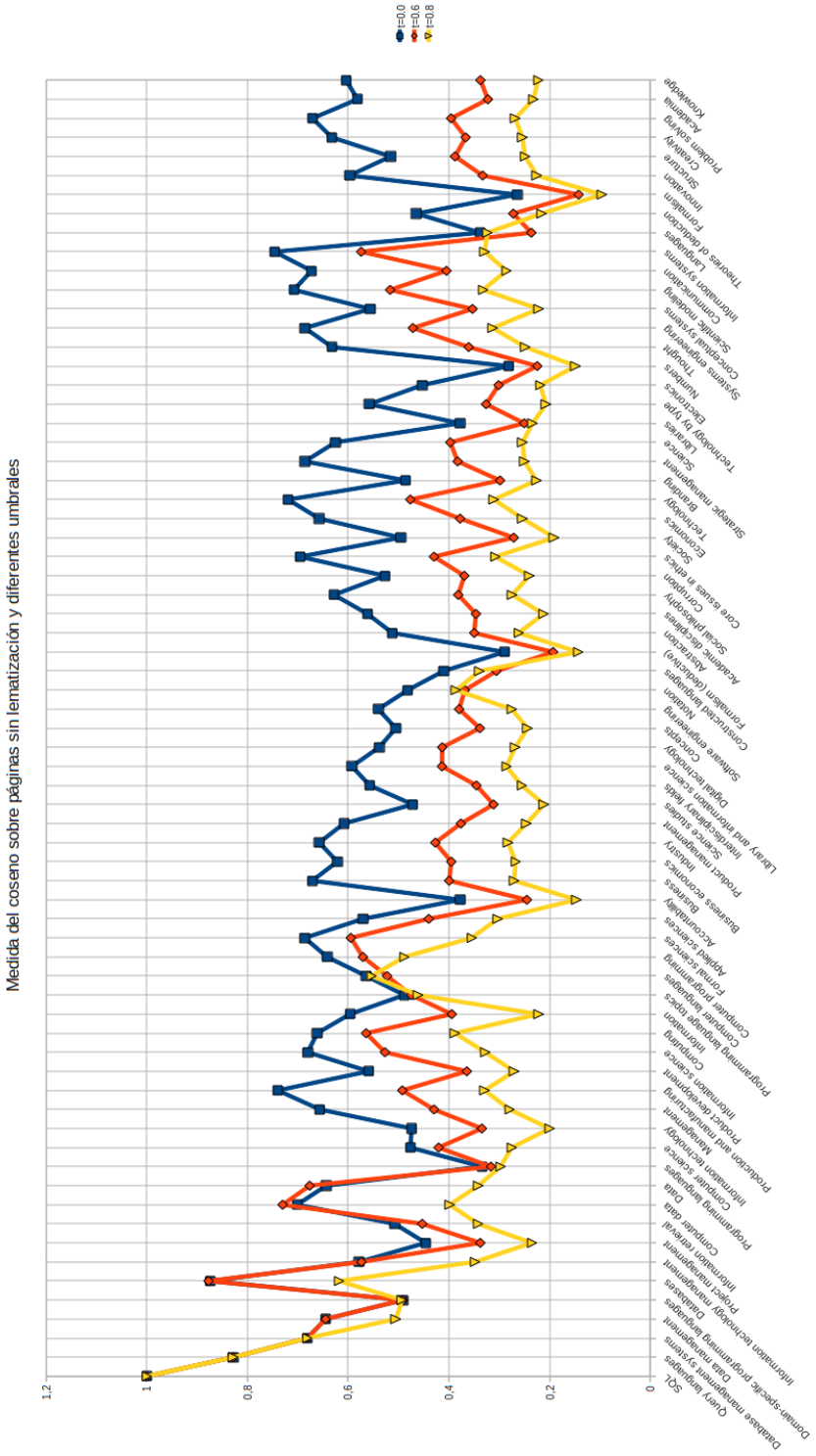


Figura 5.28: Valores obtenidos para las categorías para cada uno de los métodos

valor, más ruido iremos incluyendo en la categoría acumulada. La idea es la de aumentar el valor obtenido incluyendo nuevos conceptos relacionados en la comparación, si además rebajamos el umbral de aceptación, tendremos una nueva fuente de imprecisión, lo que puede llevarnos a una situación no deseada, en la que cada vez sea más fácil incluir nuevas categorías, lo cual provoca la obtención de una ontología final con demasiados conceptos.

El método empleando categorías acumuladas es muy versátil, ya que nos permite variar los valores obtenidos para las distintas categorías, simplemente variando el valor del umbral, desde un umbral igual a 1 (este caso sería equivalente al de comparación con la categoría inicial), hasta un valor de umbral 0.

También es posible emplear un doble umbral, el primer umbral serviría para determinar la inclusión de una categoría en el contexto (la categoría acumulada con la que comparamos), y el otro determinaría el valor mínimo que debe obtener una categoría en la evaluación para incluirse en el resultado final. Mientras el primer umbral determina el contexto de las comparaciones, el segundo determina la condición para la inclusión de una categoría en el resultado final.

5.3. Evaluación de los Métodos

En esta sección realizaremos un análisis de los datos obtenidos en la evaluación de los distintos métodos que se han presentado con anterioridad. El análisis estará basado en los datos resumidos obtenidos de la experimentación realizada. Todas las fases de la experimentación y una descripción detallada de las diferentes decisiones tomadas a lo largo del proceso de experimentación se pueden ver en el Capítulo 6.

Para poder comprobar la efectividad de las distintas medidas, es necesario realizar experimentos bajo las mismas condiciones, con el objetivo de aislar posibles factores que hagan variar la bondad de las medidas.

5.3.1. Relevancia Semántica

Un buen criterio para determinar la utilidad de un método frente a otro es el cálculo de la correlación de sus evaluaciones con juicios humanos.

Para calcular la relevancia semántica de un método, existen diferentes conjuntos de datos, que proporcionan un entorno controlado en el que poder realizar experimentos y poder comprobar los resultados obtenidos en igualdad de condiciones. La relevancia semántica se determina por comparación con conjuntos de datos anotados por humanos. La forma de comparar diversos métodos entre sí, es aplicarlos a alguno de estos conjuntos de datos y medir la correlación con los juicios realizados por humanos, cuanto mayor sea la correlación mejor será la medida.

Uno de esos conjuntos de referencia es WordSimilarity-353 [Finkelstein et al., 2002], el cual contiene 353 pares de términos y juicios de relación proporcionados por humanos. Calculando la correlación entre los valores dados por los humanos y los obtenidos por un método, podemos determinar cómo se aproxima a los criterios de relación proporcionados por humanos. Dadas las particularidades de nuestro caso, tendremos que seleccionar un subconjunto de WS-353, que llamaremos WS-209. Los detalles acerca de cómo hemos

seleccionado el conjunto y cómo se han calculado los valores de correlación se detalla en la Sección 6.3.1.

Para los distintos métodos, los valores de correlación obtenidos se muestran en la tabla 5.13. Es necesario mencionar que no es posible calcular valores de relevancia semántica para los métodos basados en exploración de grafos, ya que estos no establecen ningún tipo de evaluación y comparación de categorías, sino que se limitan a seleccionar nodos en un camino que se dirige hacia la raíz del grafo. En la tabla se muestra en negrita el método con mejor valor de correlación, y en negrita y cursiva el mejor método global, desde el punto de vista de la correlación de los valores obtenidos con juicios humanos.

Tabla 5.13: Correlación de los distintos métodos empleados

Método	Correlación WS-209
Métodos basados en Índices Estadísticos	
Media Jaccard	0.2559
Mejor Jaccard	0.2993
Media Sorensen	0.2741
Mejor Sorensen	0.3122
Media Mountford	0.2303
Mejor Mountford	0.2841
Media Overlap	0.2776
Mejor Overlap	**0.3296**
Métodos de Similitud Semántica	
Media Rada	-0.2974
Mejor Rada	-0.2942
Media Leacock-Chodorow	0.3358
Mejor Leacock-Chodorow	**0.3635**
Media Wu-Palmer	0.2764
Mejor Wu-Palmer	0.2635
Métodos de Contenido de la Información	
Media Resnik	0.2670
Mejor Resnik	0.3032
Media Lin	0.2760
Mejor Lin	**0.3113**
Media Jiang-Conrath	-0.0565
Mejor Jiang-Conrath	0.1079
Métodos de Recuperación de Información	
Media Coseno	***0.4993***
Mejor Coseno	0.4176

Podemos observar que los valores de correlación obtenidos son muy bajos. Sin embargo están dentro del rango habitual de valores obtenidos para el conjunto WordSimilarity-353. En nuestro caso, hemos empleado un subconjunto de éste, para aislar en la medida de lo posible el comportamiento de los métodos de factores externos tales como la desambiguación de términos. Además, todos los métodos han sido evaluados con respecto a la imagen de Wikipedia con la que estamos trabajando, por lo que la comparación se ha realizado en las mismas condiciones para todos.

De entre los métodos evaluados vemos como el basado en recuperación de información ofrece mayores valores de correlación. Aunque esto de por sí lo convierte en un buen candidato para ser utilizado en la extensión semántica de ontologías, vamos a ver otros criterios para la selección del método más adecuado.

Para poder comparar los distintos métodos, evaluaremos la utilidad de la ontología dentro de un aplicación. En este caso la aplicación consiste en la consulta de conceptos empleando consultas textuales. A partir de un campo textual se confecciona una ontología siguiendo la metodología descrita para la extracción de semántica a partir de texto con Wikipedia. La ontología tiene cada concepto anotado con un conjunto de términos extraídos de WordNet, que sirven para recuperar dicho concepto de un texto. Esto permite recuperar el concepto objeto de consulta en otros textos. En nuestro caso, las consultas se ejecutan sobre el campo de texto original del que se obtuvo la ontología. Esto permite calcular la cobertura del conjunto inicial, es decir, como de bien representa la ontología al texto original. Todo el proceso detallado se puede consultar en la Sección 6.3.3.

Para realizar los experimentos se han recopilado 4 conjuntos de datos, `CCIA` con 500 títulos de artículos de investigación en el área de Ciencias de la Computación, `MEDLINE` con 500 títulos de artículos de investigación en biomedicina, `SPORT` con 500 titulares deportivos extraídos de noticias de Reuters, y `MISC` con 500 tiulos seleccionados de forma aleatoria de entre los conjuntos anteriores. Una descripción mas detallada de los conjuntos de datos y los criterios seguidos para su selección se pueden encontrar en la Sección 6.1.

Utilizando valores de umbral en el intervalo $[0.6, 1]$ en incrementos de 0.025, generamos las ontologías para todos los métodos basados en medidas y para todos los conjuntos de datos.

Evaluamos los métodos realizando búsquedas sobre conceptos usando las ontologías obtenidas. Utilizaremos la cobertura del conjunto inicial (*recall*), como medida para la comparación. Por cada una de las ontologías anotaremos el número de nodos/clases, la cobertura y el tiempo de generación. Para valores superiores a 0.9 casi todos los métodos se comportan igual. Las diferencias más significativas las podemos encontrar para valores de umbral de 0.6.

Usando los valores recogidos para las ontologías generadas con umbral 0.6, realizamos un test de Friedman para determinar un orden en los algoritmos atendiendo a la cobertura. La ordenación se muestra en la Figura 5.29, valores bajos indican mejor posición y por tanto que el método es mejor. Evaluaremos todos los métodos usando las posibles variantes con respecto al ámbito de comparación, de este modo tendremos métodos usando comparación con la categoría inicial (1), la categoría actual (2) y la categoría acumulada (3). Las tablas completas para los valores de ranking, calculados para distintos umbrales, se pueden consultar en la Sección 6.5.

El test de Friedman señala el método de WU-PALMER-2 como el mejor. Para ver si realmente la diferencia con el resto de métodos es significativa realizamos el test *post hoc* de Hommel. Los p-valores ajustados se muestran en la Tabla 5.14.

El test *post hoc* determina la existencia de diferencias significativas con respecto a WU-PALMER-2 para todos los métodos excepto para LIN-2, COSENO y JIANG-CONRATH en todas sus versiones y WU-PALMER-2. Es decir, no es posible estadísticamente decir que uno de ellos es mejor que los otros.

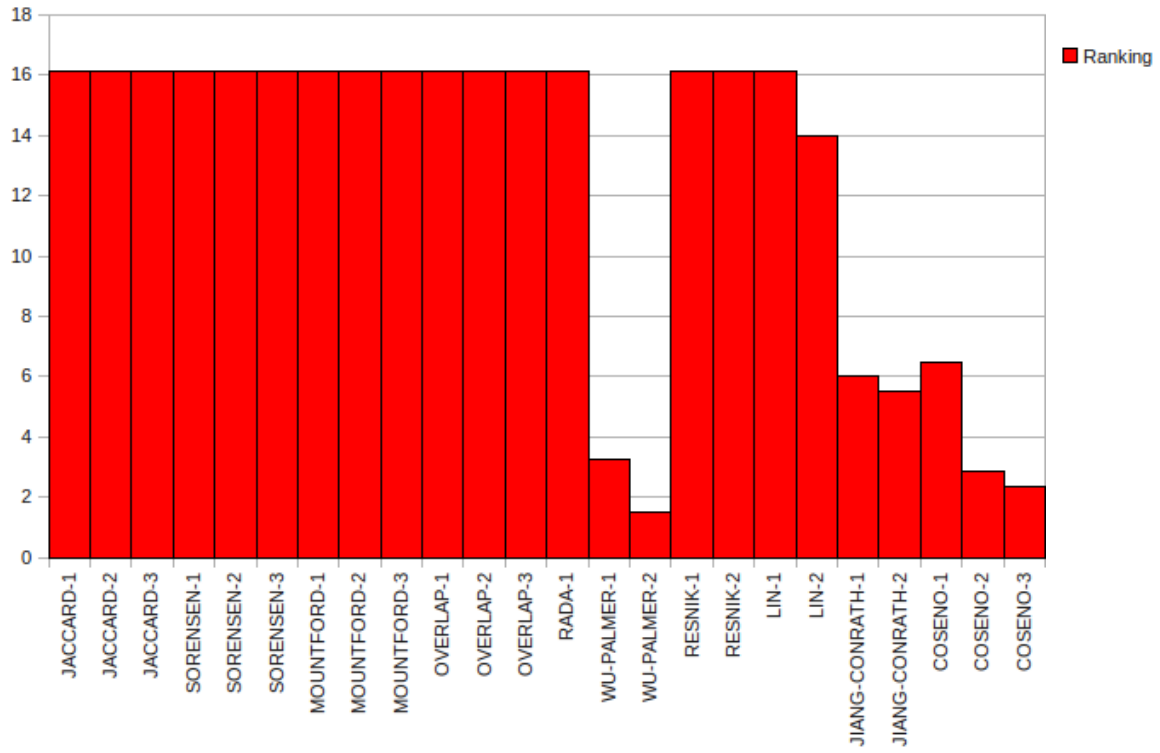


Figura 5.29: Ranking para los métodos (Friedman)

Tabla 5.14: p-valores ajustados (Friedman)

i	algoritmo	p sin ajustar	p_{Hommel}
1	JACCARD-1	0.003445	0.027556
2	JACCARD-2	0.003445	0.027556
3	JACCARD-3	0.003445	0.027556
4	SORENSEN-1	0.003445	0.027556
5	SORENSEN-2	0.003445	0.027556
6	SORENSEN-3	0.003445	0.027556
7	MOUNTFORD-1	0.003445	0.027556
8	MOUNTFORD-2	0.003445	0.027556
9	MOUNTFORD-3	0.003445	0.027556
10	OVERLAP-1	0.003445	0.027556
11	OVERLAP-2	0.003445	0.027556
12	OVERLAP-3	0.003445	0.027556
13	RADA-1	0.003445	0.027556
14	RESNIK-1	0.003445	0.027556
15	RESNIK-2	0.003445	0.027556
16	LIN-1	0.003445	0.027556
17	LIN-2	0.012419	0.086935
18	COSENO-1	0.317311	0.86108
19	JIANG-CONRATH-1	0.36812	0.86108
20	JIANG-CONRATH-2	0.423711	0.86108
21	WU-PALMER-1	0.726339	0.86108
22	COSENO-2	0.783316	0.86108
23	COSENO-3	0.86108	0.86108

Así que para determinar el mejor entre los seleccionados debemos remitirnos a otras características. Si tenemos en cuenta la correlación será mejor usar el COSENO, si queremos que el tiempo de ejecución sea bajo podemos usar JIANG-CONRATH y si queremos obtener ontologías con muchas clases y que no tengan un tiempo de ejecución excesivo podemos usar WU-PALMER.

La selección del método a utilizar en cada caso dependerá de la aplicación considerada, ya que cada aplicación tiene sus propios requisitos y por tanto el uso de un método u otro dependerá de los detalles específicos de cada problema. El análisis pormenorizado de los resultados se puede ver en la Sección 6.5.

5.4. Conclusiones

En este capítulo hemos visto cómo, a partir de textos no estructurados y mediante un proceso de minería de datos, podemos obtener una representación estructurada en forma de Conjuntos-AP y obtener semántica para estos a través de Wikipedia.

La utilización de un diccionario lexicográfico como es WordNet nos permite obtener significado para los términos textuales que estamos procesando. Obtener una representación para todas y cada una de las palabras presentes en el texto no parece lo más adecuado, puesto que la ontología obtenida sería muy grande y poco representativa. Sin embargo, gracias a la representación de textos breves basado en Conjuntos-AP, podemos determinar los conceptos mas relevantes debido a su frecuencia de aparición, y con la ayuda de Wikipedia, generar una ontología más compacta que capture los principales conceptos que componen el dominio.

Como se vio en el capítulo anterior, el uso de WordNet tiene como principal inconveniente su excesiva orientación lingüística y falta de vocabulario de dominio específico para determinadas materias. Con el objetivo de solventar estas carencias reemplazamos el uso de WordNet por el de Wikipedia para la extensión semántica de la estructura de representación intermedia. Wikipedia tiene una estructura menos refinada que la de WordNet, pero está en constante evolución y contiene una amplia variedad de terminología. Mediante el estudio de diversas métricas de similitud y relación semántica hemos presentado una metodología para la generación de una ontología a partir del grafo de categorías de Wikipedia que represente de forma adecuada la Estructura-AP obtenida para una columna de contenido textual en una base de datos.

Si bien las ontologías generadas a partir de Wikipedia son más completas, toda la información acerca de sinónimos sólo podemos extraerla de WordNet, por lo que combinaremos ambas aproximaciones con el objetivo de aprovechar las ventajas de cada una de ellas. Mientras toda la etapa de procesado semántico se realiza con WordNet, la etapa de generación de la ontología a partir de la Estructura-AP se realiza teniendo en cuenta el grafo de categorías de Wikipedia, para finalmente ampliar cada una de las clases generadas con los conjuntos de sinónimos de los que disponga WordNet.

Particularmente en esta etapa en la que nos encontramos, estamos centrándonos más en la obtención de una taxonomía que ofrezca un punto de partida para enriquecer el modelo ontológico generado. Como trabajos futuros está pendiente la inclusión de nuevas

relaciones semánticas en la ontología, extraídas de WordNet u otra fuente de contenido semántico.

Una vez construida la ontología que define el dominio de la columna de texto, vamos a crear una herramienta a modo de Plug-In de Protégé que permita consultar los textos a través de la ontología obtenida.

También estamos estudiando la forma de integrar la ontología de dominio obtenida para la columna, con la ontología que describe la estructura del esquema de la base de datos, tal como se vio en el Capítulo 3.

Capítulo 6

Evaluación Experimental de los Métodos

En el capítulo anterior hemos realizado diferentes propuestas para la selección de categorías relacionadas con términos lingüísticos para la creación de una ontología descriptiva de dichos términos. Hemos presentado diferentes métodos y medidas para la selección de las categorías, y se ha realizado una breve argumentación acerca de los resultados obtenidos.

En este capítulo vamos a realizar una evaluación cuantitativa extensiva de cada uno de estos métodos y medidas, con el fin de seleccionar el más adecuado. Para ello vamos a recopilar diversos conjuntos de datos de distintos temas y vamos a aplicar los métodos a dichos conjuntos. Evaluaremos diferentes aspectos de los resultados obtenidos y finalmente realizaremos una propuesta de método, con el que mejor resultados obtenga en las diferentes evaluaciones.

6.1. Descripción de los Conjuntos de datos

Para la realización de las pruebas hemos confeccionado cuatro conjuntos de datos, cada uno de ellos con sus características particulares, con la finalidad de comprobar el comportamiento de los distintos métodos ante conjuntos de datos de diversa procedencia. Estableceremos cuatro escenarios diferentes, y veremos cómo se comportan cada uno de los métodos, seleccionando finalmente aquellos cuyos resultados obtenidos sean los más prometedores.

6.1.1. Conjunto 1 : CCIA-500

En este conjunto de datos hemos incluido 500 títulos de artículos científicos realizados por miembros del departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada. Todos los títulos contenidos en este conjunto de datos están en inglés y tratan acerca de diversas disciplinas relacionadas con las ciencias de la computación.

Este conjunto contiene terminología técnica específica y escasa presencia de entidades con nombre. Existe una cierta predominancia de determinados temas, atendiendo a la actividad investigadora de los distintos grupos pertenecientes al departamento. Por lo tanto, aunque el posible espectro temático en ciencias de la computación es muy amplio, en este caso está focalizado en cada una de las líneas de investigación llevadas a cabo en el mencionado departamento.

Al conjunto se le han aplicado las dos etapas de la metodología dedicadas al preprocesamiento. En la etapa de preprocesamiento sintáctico se ha realizado tokenización, eliminación de palabras vacías y se ha aplicado un filtro para eliminación de plurales. El preprocesamiento semántico se ha realizado utilizando WordNet.

Para la generación de los Conjuntos-AP en este conjunto de datos se ha empleado un soporte de 0.015.

6.1.2. Conjunto 2 : MEDLINE-500

El segundo conjunto consta de una selección de 500 títulos de artículos procedentes de MEDLINE, una base de datos en línea en inglés, con más de 11 millones de citas y resúmenes de artículos en revistas médicas. En este caso la selección se ha realizado de forma aleatoria. Este conjunto, a pesar de tener una gran cantidad de vocabulario técnico como ocurría con el *Conjunto 1*, posee una variedad temática más amplia puesto que no hemos establecido condiciones en la selección de los títulos. Este conjunto de datos contiene algunas entidades con nombre, como pueden ser nombres de genes, proteínas, etc...

En este caso también hemos realizado ambas etapas de preprocesamiento, y éstas se han realizado con la misma configuración que en el conjunto anterior. La generación de los conjuntos-AP para este conjunto de datos se ha realizado utilizando un soporte de 0.010. En este caso se ha bajado un poco el soporte puesto que el contenido es más general y no se centra en aspectos específicos dentro del campo médico, sino que se han obtenido registros al azar.

6.1.3. Conjunto 3 : SPORT-500

Este conjunto posee un vocabulario técnico, pero ampliamente difundido, como es el vocabulario del mundo del deporte. Además, incluye una elevada cantidad de entidades con nombre, tales como pueden ser nombres de equipos, de competiciones y deportistas. La selección se ha realizado sobre los titulares deportivos suministrados por REUTERS en su división americana, para los meses de Noviembre y Diciembre de 2010. Este conjunto de datos introduce una nueva variable que es interesante observar, la velocidad con la que el nuevo conocimiento se integra en herramientas de carácter colaborativo en línea, como Wikipedia, frente a otros modelos más estáticos como puede ser el de WordNet.

Aunque la etapa de preprocesamiento semántico se ha mantenido, se han introducido variantes en la etapa de preprocesamiento sintáctico. En esta ocasión, debemos de tener en cuenta que este conjunto contiene una elevada cantidad de entidades con nombre. En algunos casos, estas entidades están compuestas por términos que han de ser desambiguados. Algunas etapas del preprocesamiento sintáctico empleado anteriormente pueden desvirtuar

estas entidades. Por ejemplo, la eliminación de plurales puede cambiar ciertos nombres de equipos *Raptors*, *Celtics*, etc... Es por esto, que sólo realizaremos tokenización y eliminación de palabras vacías.

De nuevo, al tratarse éste de un conjunto de datos genérico sin un dominio específico, la generación de los conjuntos-AP para este conjunto de datos se ha realizado utilizando un soporte de 0.010.

6.1.4. Conjunto 4 : MISC-500

Este conjunto contiene una selección aleatoria uniforme de 500 datos elegidos de entre los tres conjuntos anteriores. Frente a los tres conjuntos anteriores que mantenían una coherencia temática, éste es mas disperso e incluye temas bastante alejados entre sí. Esto nos permitirá ver si los métodos empleados facilitan la integración temática, o si por el contrario la dispersión fomentará la falta de información.

Ya que el conjunto incluye datos de diversa procedencia, y dado que no podemos establecer de forma clara el porcentaje de entidades con nombre que forman parte del conjunto, hemos optado por realizar un procesamiento homogéneo para los datos. Así pues, el preprocesamiento sintáctico incluye la tokenización, eliminación de palabras vacías y la eliminación de plurales, y el semántico se ha realizado utilizando WordNet.

En esta ocasión, dado que este conjunto de datos surge de la unión de los diferentes conjuntos de datos, y por tanto, es heterogéneo, hemos fijado el valor para el soporte en 0.010 para la generación de los conjuntos-AP. Empleamos el valor mínimo de soporte de los conjuntos incluidos.

En la Tabla 6.1 mostramos las estadísticas obtenidas en el proceso de limpieza semántica y en la generación de Conjuntos-AP.

Tabla 6.1: Estadísticas para el preprocesamiento y la generación de Conjuntos-AP

Medida	**CCIA-500**	**MEDLINE-500**	**SPORT-500**	**MISC-500**
Estadísticas del Preprocesamiento Semántico				
Palabras Procesadas	3420	5612	3170	4130
Palabras Reconocidas	3250	4641	2584	3525
Palabras No Reconocidas	156	730	473	463
Palabras Eliminadas en POS	14	241	113	142
Total de Cambios Registrados	309	623	786	543
Estadísticas de la Generación de Conjuntos-AP				
Tuplas procesadas	500	500	500	500
Palabras Perdidas	2054	4535	2502	3482
Porcentaje de Palabras Perdidas	60.05 %	80.80 %	78.92 %	84.30 %

6.2. Evaluación

Con el objetivo de determinar las medidas y métodos mas adecuados para utilizar en nuestros algoritmos, debemos realizar una evaluación de las distintas opciones y decantarnos por aquella que sea más adecuada a nuestras necesidades. La evaluación de las

medidas y métodos se realizará en dos vertientes distintas. La primera evaluación se realizará desde un punto de vista meramente semántico, comprobando la relevancia semántica de las medidas empleadas. Por otra parte, realizaremos una evaluación de los resultados obtenidos. Dichos resultados son ontologías, las cuales han de ser evaluadas para determinar su idoneidad. Antes de comenzar a describir los experimentos a realizar, es necesario hacer un breve repaso a todo lo relacionado con la evaluación de ontologías.

La evaluación de ontologías consiste en determinar la validez de una ontología para un uso determinado. Normalmente se emplea para determinar cual, de entre una serie de ontologías, es mejor para un determinado propósito.

A pesar de que se lleva trabajando en evaluación de ontologías desde 1994, aún no existe consenso sobre un método detallado y global que permita resolver este problema. Particularmente es complicado encontrar metodologías para la evaluación automática de ontologías. A continuación veremos algunas de las aproximaciones al problema de la evaluación de ontologías realizado por diversos autores, y nos centraremos particularmente en aquellas técnicas válidas para realizar evaluaciones automáticas.

El proceso de evaluación de una ontología no es una tarea claramente definida, atendiendo a distintos aspectos de la ontología y su aplicación se pueden utilizar diferentes métodos. En [Brank et al., 2005] se realiza una clasificación de las distintos métodos de evaluación en cuatro grupos:

- Comparación con un estándar.
- Utilización de la ontología en una aplicación concreta y evaluación de los resultados obtenidos.
- Comparación con una fuente de datos en cuanto al dominio que cubre la ontología.
- Evaluación realizada por expertos humanos.

Aparte de esta categorización inicial también incide en la importancia de agrupar los distintos métodos atendiendo al nivel en el que realiza la evaluación.

La evaluación puede realizarse a los siguientes niveles:

- *Léxico, vocabulario y nivel de datos:* Se evalúan conceptos, instancias, etc... Se mide la distancia entre los componentes léxicos de la ontología con respecto a un estándar. Esta medida se puede realizar con medidas de distancia o adaptando conceptos de recuperación de información como *precision* y *recall*. En ambos casos es necesario un estándar con el que comparar.
- *Jerarquía o taxonomía:* Se evalúan las relaciones entre conceptos. Dicha evaluación en este caso también se realiza por comparación con un estándar.
- *Nivel de contexto o aplicación:* La evaluación en estos casos consiste en emplear las distintas ontologías sobre la aplicación y evaluar sus resultados. La evaluación de los resultados, puede ser mucho más intuitiva y directa que la evaluación de la ontología en sí. El problema radica en que para comparar dos ontologías ambas deben ser aptas para el uso en la misma aplicación.

- *Nivel sintáctico:* La evaluación a este nivel se realiza cuando las ontologías se han construido de forma manual.
- *Estructura, arquitectura y diseño:* Como en el caso anterior, la evaluación a este nivel está mas orientada a un caso en el que la ontología se construye manualmente y se emplean una serie de principios de diseño establecidos.

De forma similar en [Vrandecic, 2009] se analizan los distintos aspectos de una ontología que son susceptibles de ser evaluados:

- *Vocabulario:* El conjunto de nombres utilizado en la ontología.
- *Sintaxis:* Trata acerca de las distintas posibilidades de serialización de las diferentes sintaxis.
- *Estructura:* Se evalúa la ontología en base al grafo subyacente que la define.
- *Semántica:* Se corresponde con la interpretación de la ontología y el conjunto de modelos que describe.
- *Representación:* Relación entre la estructura y la semántica.
- *Contexto:* Este aspecto está relacionado con la comparación de la ontología con otros elementos dentro del mismo contexto, tal como puede ser una aplicación que usa la ontología.

En nuestro caso particular la evaluación por parte de expertos o la comparación con un estándar no parecen ser las opciones más adecuadas. Dado el tamaño de las ontologías que se van a obtener y las mínimas diferencias estructurales entre ellas, la tarea implicaría un alto conocimiento del experto del dominio textual a partir del cual se obtienen las ontologías, por no mencionar la cantidad de tiempo necesario para evaluar manualmente la ontología. Por otra parte, la comparación con un estándar tampoco es una opción factible. No existen ontologías de referencia para todos los temas posibles sobre los que van a tratar las ontologías de salida, ya que éstas van a modelar la semántica de un conjunto de textos y por tanto dependerán de éste.

La evaluación de la ontología en base a su utilidad en el contexto de una aplicación concreta, parece ser un mejor método de evaluación. Dado que la ontología de salida se va a utilizar para extender consultas de usuario sobre los datos textuales originales, podemos probar las diferentes ontologías sobre esta aplicación de recuperación de información y evaluar los resultados obtenidos. De igual modo, como la ontología se genera a partir de un conjunto de textos, podemos evaluar el dominio cubierto por la ontología frente al cubierto por los datos originales. Tal y como hemos visto anteriormente, el dominio de la ontología será más reducido que el original dadas las características del proceso de construcción de la ontología. Pero el objetivo en este caso, no es tanto evaluar la ontología frente al conjunto de datos original, sino evaluar unas ontologías contra otras teniendo como referencia ese conjunto de datos.

Con respecto a los niveles a los que vamos a evaluar, atendiendo a los métodos de evaluación que consideramos adecuados a nuestro caso, podemos realizar una evaluación a nivel de contexto o aplicación. Adicionalmente también es posible realizar un análisis a nivel estructural, pero necesitaremos de medidas que nos proporcionen valores que se puedan comparar.

Para profundizar en este aspecto nos dirigimos a [Gangemi et al., 2006], donde se realiza una clasificación de medidas de evaluación atendiendo al nivel al que se van a aplicar.

- *Medidas estructurales:* Se basan en la sintaxis y semántica formal típicas de las ontologías representadas como grafos.
- *Medidas funcionales:* Relacionadas con el uso que se va a hacer de la ontología y sus componentes dentro de un contexto.
- *Medidas de usabilidad:* Dependientes del nivel de anotación de la ontología considerada.

En nuestro caso nos interesan principalmente las medidas estructurales y funcionales. En [Gangemi et al., 2006] se proponen una amplia variedad de medidas, el total de las cuales asciende a casi 40. Muchas de estas medidas presentan cierta redundancia entre sí, por lo que únicamente presentaremos aquí aquellas que hayamos seleccionado para aplicar en nuestra evaluación.

6.3. Descripción de los experimentos

Para poder evaluar la bondad de cada uno de los métodos propuestos vamos a proponer una serie de experimentos que nos van a permitir obtener medidas que evalúen distintas características. Para determinar si los valores que hemos obtenido son apropiados, debemos justificar que las medidas empleadas se corresponden con criterios similares a los que emplearía un experto humano, que los resultados son manejables y que son representativos, en la medida de lo posible, del conjunto de datos inicial a partir del cual se han generado.

Desglosando estos requisitos vemos que podemos realizar tres tipos diferentes de evaluación :

- **Evaluación de la relevancia semántica en los criterios de selección de categorías:** Para evaluar cada método en esta categoría realizaremos la evaluación del conjunto WordSimilarity-353 para cada método y calcularemos los valores de correlación correspondientes. El método con mejor correlación será por tanto, aquel cuyos valores de relación sean más cercanos a los que ofrecería un experto.
- **Evaluación estructural:** A través de la evaluación de la estructura de la ontología seleccionaremos aquellas cuyo número de nodos y estructura sea apropiada para una adecuada comprensión. Para ello tomaremos medidas acerca del número de nodos obtenidos, profundidad máxima de la jerarquía, número máximo, medio y mínimo de padres por nodo, número máximo, medio y mínimo de hermanos por nodo, etc...

- **Cobertura del conjunto de datos inicial:** Dado que la ontología obtenida debe representar el dominio subyacente de la forma más apropiada y proporcionar términos de consulta que permitan recuperar el conjunto de datos sobre el que se ha construido, la medida de cobertura del conjunto de resultados inicial, supone un indicador de en qué medida el dominio está realmente bien representado.

6.3.1. Evaluación de la Relevancia Semántica

Uno de los factores que necesitamos tener en cuenta es, cómo de apropiada es la medida que empleamos en la selección de categorías. Para nosotros la medida ideal sería aquella que produjese unas evaluaciones lo más parecidas a las realizadas por un humano. Con los juicios emitidos por la medida correspondiente con respecto a la relación entre dos categorías, se puede entonces seleccionar nuevas categorías en base a otras confiando en que el criterio elegido es el adecuado.

Para comparar medidas de similitud y relación semántica, existen tres conjuntos de datos ampliamente usados que están disponibles de forma pública.

En [Rubenstein and Goodenough, 1965] se presentó el primer conjunto de datos para medir la similitud semántica entre un par de conceptos. En este caso, se tomaron 65 pares de nombres y se entregaron a un total de 51 sujetos en diferentes sesiones, para que juzgasen con una puntuación entre 0 y 4 el grado de similitud entre el par de nombres. Posteriormente se realizó la media de las apreciaciones realizadas por los sujetos, y se estableció un valor numérico para cada uno de los pares.

En [Miller and Charles, 1991] se presentó otro conocido conjunto de datos de referencia. En esta ocasión el conjunto de pares eran 30, y eran un subconjunto de los pares R&G.

El conjunto de pruebas WordSimilarity-353 [Finkelstein et al., 2002] contiene dos conjuntos de palabras en inglés junto con juicios de relación asignados por sujetos humanos. El primer conjunto contiene 153 pares de palabras junto con sus evaluaciones de relación asignados por 13 sujetos, y las medias de las evaluaciones. El segundo conjunto contiene 200 pares de palabras, con el juicio de relación realizado por 16 sujetos y sus correspondientes medias. Las estimaciones se encuentran en el rango de 0 a 10, variando desde la total independencia a la identidad. Existe un conjunto de pruebas combinado que es la concatenación de los dos anteriores. El primer conjunto contiene los 30 pares de M&C, pero la evaluación se ha vuelto a realizar desde cero.

Mientras R&G y M&C contienen valores de similitud semántica, WS está evaluado con criterios de relación semántica.

Para nuestra evaluación utilizaremos el conjunto WordSimilarity-353, puesto que posee un mayor número de pares de términos evaluados, y porque contiene términos de los otros conjuntos de datos.

El conjunto posee evaluaciones sobre pares de términos, pero en nuestro caso la evaluación la vamos a realizar sobre categorías. El proceso por el cual se obtienen las categorías correspondientes a un término, consiste en determinar la página más adecuada para el término y a partir de ella obtener las categorías asociadas a la página, y por tanto, al término. Este proceso introduce una fuente de ambigüedad, ya que debemos determinar el sentido del término para poder recuperar la página adecuada.

Dado que nosotros trabajaremos directamente sobre categorías, el ruido introducido por términos no desambiguados correctamente, podría desvirtuar los resultados que vamos a obtener en la evaluación de las medidas de relevancia semántica. Para evitar este inconveniente, evaluaremos las medidas únicamente con aquellos pares de términos que no presentan ambigüedad. Tomando todos los pares del conjunto original y eliminando aquellos términos ambiguos, obtenemos un conjunto de 209 pares. Será sobre este conjunto sobre el que evaluamos la relevancia semántica de las medidas. Denominaremos a este conjunto de pares no ambiguos WS-209.

Aunque hemos podido aislar el factor ambigüedad con la selección de los términos, aún tenemos un factor que hará variar los resultados de forma significativa. Una vez se ha determinado la página correspondiente a un término, ésta puede estar categorizada en una o varias categorías, ¿cual de estas escogemos para la evaluación?. Debemos de tener en cuenta que si la página forma parte de varias categorías, cada una de éstas aporta una información semántica distinta. Así, en lugar de evaluar una única categoría, evaluaremos todas las categorías y obtendremos dos valores, la media de todas las evaluaciones, y el valor más alto de las evaluaciones realizadas.

Para nuestras pruebas hemos tomado el conjunto de 209 pares, y hemos evaluado las distintas medidas. Con los resultados obtenidos, y aplicando el coeficiente de correlación de Pearson, obtenemos una valoración de la correlación existente entre ambos conjuntos de datos. El coeficiente de Pearson estudia la fuerza de la relación lineal entre dos variables cuantitativas y ofrece valores en el rango [-1,1]. Un valor de 1 indica una relación lineal positiva total y 0 ninguna correlación. Los valores negativos indica el valor de correlación negativo entre las muestras.

En la tabla 6.2 podemos ver la interpretación para los valores de correlación de Pearson.

Tabla 6.2: Interpretación del valor de correlación de Pearson

Valor de Correlación	Interpretación
0.0	No hay correlación
0.3	Baja correlación
0.5	Media correlación
0.7	Alta correlación
1	Correlación perfecta

6.3.2. Evaluación estructural

En el caso de la evaluación estructural, nos encontramos con el problema de decidir cual es la estructura adecuada de una ontología en cada caso. Dado que la tarea no es fácil puesto que depende de muchos factores, centraremos la evaluación estructural en ofrecer una serie de medidas absolutas de nos permitan describir de la forma más completa posible la ontología. Aunque no será posible comparar los distintos métodos con los valores obtenidos, las medidas son de utilidad para describir el tipo de ontologías que se genera con cada una de las distintas aproximaciones evaluadas.

De entre las posibles medidas presentadas en [Gangemi et al., 2006] seleccionamos aquellas que nos pueden ser de utilidad. Éstas se definen a continuación.

Medidas

Teniendo en cuenta las siguientes definiciones:

Sea $G = (V, A, r)$ un grafo G con un conjunto de nodos V, un conjunto de arcos A y un nodo raíz r.

Definición 6. *Definimos el conjunto de nodos hoja $LEA \subseteq V$, siendo aquellos que no tienen nodos hijos en el grafo G, o lo que es lo mismo, aquellos que no tienen arcos de tipo es-un apuntando hacia ellos $\forall LEA \in V \nexists a \in A | ter(a) = LEA_i$.*

Definición 7. *Se define el conjunto de nodos padre $PAR_{i \in V}$ como el conjunto de nodos $PAR \subseteq V$ conectados a un mismo nodo hijo i de G a través de un arco es-un. $PAR_i = \forall V | init(a) = i$.*

Definición 8. *Se define el conjunto de nodos hermanos $SIB_{j \in V}$ como el conjunto de nodos $SIB \subseteq V$ conectados al mismo nodo j de G a través de un arco es-un. $SIB_j = \forall V | ter(a) = j$.*

Comenzaremos recopilando valores estadísticos simples tales como:

- **Número de nodos.**

$$m = n_G = |V| \tag{6.1}$$

- **Número máximo de padres por nodo.**

$$\begin{gathered} m = N_{j \in PAR}; \\ \forall i \exists j (N_{j \in PAR} \geq N_{i \in PAR}) \end{gathered} \tag{6.2}$$

  donde $N_{i \in PAR}$ y $N_{j \in PAR}$ son las cardinalidades de un conjunto de padres i o j pertenecientes al conjunto de padres PAR en un grafo G.

- **Número medio de padres por nodo.**

$$m = \frac{\sum_{j}^{PAR} N_{j \in PAR}}{n_{PAR}} \tag{6.3}$$

  donde la sumatoria representa la cardinalidad absoluta de hermanos en el grafo G, y n_{PAR} es la cardinalidad del conjunto PAR de G.

- **Moda del número de padres por nodo.**

$$m = mode(N_{j \in PAR}) \tag{6.4}$$

  donde $mode()$ representa la moda estadística y $N_{j \in PAR}$ la cardianlidad de un conjunto j de nodos padres.

- **Número máximo de hermanos por nodo.**

$$\begin{gathered} m = N_{j \in SIB}; \\ \forall i \exists j (N_{j \in SIB} \geq N_{i \in SIB}) \end{gathered} \tag{6.5}$$

donde $N_{i \in SIB}$ y $N_{j \in SIB}$ son las cardinalidad de un conjunto de hermanos i o j del conjunto de hermanos SIB en un grafo G.

- **Número medio de hermanos por nodo.**

$$m = \frac{\sum_{j}^{SIB} N_{j \in SIB}}{n_{SIB}} \tag{6.6}$$

donde la sumatoria representa la cardinalidad absoluta de hermanos en el grafo G, y n_{SIB} es la cardinalidad del conjunto SIB de G.

- **Moda del número de hermanos por nodo.**

$$m = mode(N_{j \in SIB}) \tag{6.7}$$

donde $mode()$ representa la moda estadística y $N_{j \in SIB}$ la cardianlidad de un conjunto j de nodos hermanos.

- **Número de nodos hoja.**

$$m = n_{LEA} = |LEA| \tag{6.8}$$

- **Profundidad máxima.**

$$\begin{gathered} m = N_{j \in P}; \\ \forall i \exists j (N_{j \in P} \geq N_{i \in P}) \end{gathered} \tag{6.9}$$

donde $N_{i \in P}$ y $N_{j \in P}$ son la cardinalidad de los nodos en un camino i o j del conjunto de caminos P en un grafo G.

- **Profundidad media.**

$$m = \frac{1}{n_{P \subseteq G}} \sum_{j}^{P} N_{j \in P} \tag{6.10}$$

donde $N_{j \in P}$ es la cardinalidad de cada camino j del conjunto de caminos P en el grafo G, y $n_{P \subseteq G}$ es la cardinalidad de P. En definitiva se trata de encontrar la longitud de cada uno de los posibles caminos desde la raíz a las hojas, entre el número total de caminos.

- **Anchura máxima.**

$$\begin{gathered} m = N_{j \in P}; \\ \forall i \exists j (N_{j \in P} \geq N_{i \in P}) \end{gathered} \tag{6.11}$$

donde $N_{i \in P}$ y $N_{j \in P}$ son la cardinalidad de cualquier generación i o j del conjunto de niveles L en un grafo G.

- **Anchura media.**

$$m = \frac{1}{n_{L \subseteq G}} \sum_{j}^{L} N_{j \in L} \tag{6.12}$$

donde $N_{j \in L}$ es la cardinalidad de cada generación j del conjunto de generaciones L en un grafo dirigido G, y $n_{L \subseteq G}$ la cardinalidad de L.

- **Complejidad (Tangledness)**

$$m = \frac{n_G}{t \in V \wedge \exists a_1, a,_2 \left(isa(m, a_1) \wedge (isa(m, a_2))\right)} \tag{6.13}$$

donde n_G es el número de nodos de G y $t \in V \wedge \exists a_1, a_2(isa(m, a_1) \wedge (isa(m, a_2)))$ es la cardinalidad del conjunto de nodos con más de un arco es-un (isa) saliente.

Esta medida tiene en cuenta que un nodo tenga más de un padre, por tanto, la ontología debe permitir herencia múltiple. Como se puede observar por la fórmula, el rango de la medida va desde 1 que índica el mayor valor de complejidad y tiende hasta infinito cuyo valor indica una menor complejidad. Dado que esta interpretación no es intuitiva, utilizaremos la reformulación que se hace en [Yu et al., 2009], donde se expresa la fórmula como:

$$m = \frac{t \in V \wedge \exists a_1, a,_2 \left(isa(m, a_1) \wedge (isa(m, a_2))\right)}{n_G}. \tag{6.14}$$

De este modo el valor de complejidad está normalizado en el rango $[0, 1]$ y se entiende que un valor de 0 indica baja o ninguna complejidad, mientras un valor de 1 indica alta complejidad debido a que todo nodo tiene dos o más padres.

- **Ratio de dispersión (fan-outness) en los nodos hoja.**

$$m = \frac{n_{LEA \subseteq V}}{n_G} \tag{6.15}$$

donde $n_{LEA \subseteq V}$ es la cardinalidad del conjunto de los nodos hoja y n_G es la cardinalidad del conjunto de nodos V.

- **Dispersión máxima en los nodos hoja.**

$$\begin{gathered} m = N_{j \in SIB}^{j \in LEA}; \\ \forall i \exists j (N_{j \in SIB}^{j \in LEA} \geq N_{i \in SIB}^{i \in LEA}) \end{gathered} \tag{6.16}$$

donde $N_{j \in SIB}$ y $N_{j \in SIB}$ son la cardinalidad de un conjunto de hermanos i o j que son nodos hoja, del conjunto de hermanos SIB del grafo G.

- **Ratio de dispersión en los nodos hermanos.**

$$m = \frac{\sum_{j}^{SIB} N_{j \in SIB}}{n_G} \tag{6.17}$$

donde la sumatoria representa la cardinalidad absoluta de hermanos en el grafo G, la suma de todos los hermanos presentes en el grafo, y n_G es la cardinalidad del conjunto de nodos V.

Estas medidas absolutas nos pueden ayudar a hacernos una idea aproximada de la estructura general de la ontología en cuanto a sus características estructurales.

6.3.3. Cobertura del conjunto de datos inicial

En este caso vamos a realizar una evaluación similar a la basada en aplicaciones. Dado que hemos obtenido la ontología a partir de un conjunto de textos, comprobaremos como de bien representa la ontología a dichos textos. La ontología ha sido enriquecida con términos adicionales por cada concepto, de tal forma que las consultas realizadas utilizando los conceptos se extienden con dichos términos adicionales. La aplicación que emplearemos para la evaluación, será una aplicación de consulta sobre los textos originales empleando la ontología enriquecida. Cuanto mayor sea la cobertura del conjunto de datos inicial, mejor representará la ontología a dicho conjunto.

Los medida de cobertura debe expresarse como con un valor en el intervalo $[0, 1]$ donde 0 se corresponde con la cobertura mínima y 1 con la cobertura máxima. En nuestro caso utilizaremos el *recall* como medida de cobertura, dado que consideramos todo el conjunto de datos inicial como relevante. Así pues la medida queda expresada como el cardinal del conjunto de elementos recuperados partido por el cardinal del conjunto inicial.

$$recall = \frac{N^o \text{ de elementos relevantes recuperados}}{N^o \text{ de elementos relevantes en la colección}} \tag{6.18}$$

Para determinar de forma aproximada la bondad de la cobertura en cada uno de los casos, vamos a realizar medidas de referencia base con las que comparar. La primera medida la realizaremos directamente sobre los términos obtenidos en la Estructura-AP. Debemos tener en cuenta que este valor es ajeno a la semántica obtenida para los textos, y por tanto puede ofrecer valores altos al poseer términos que se han identificado como frecuentes, pero no se ha demostrado que describan un concepto. La otra medida consiste en realizar la correspondencia conceptual con Wikipedia, pero sin expandir los nodos, de modo que tendremos una ontología de un único nivel que contendrá los términos de la Estructura-AP que se corresponden con conceptos válidos identificados en Wikipedia. La última variante es equivalente a la anterior, pero se realiza una extensión de consulta añadiendo a cada concepto los términos sinónimos asociados a éste en WordNet. Podemos consultar los valores de cobertura inicial para las medidas base en la Tabla 6.3.

Tabla 6.3: Valores base de cobertura del conjunto inicial

Método	CCIA-500	MEDLINE-500	SPORT-500	MISC-500
Texto AP-Estructura	0.892	0.89	0.76	0.67
Número de Nodos	26	35	22	24
Conceptos Wikipedia	0.674	0.616	0.448	0.338
Conceptos Wikipedia+WordNet	0.696	0.638	0.46	0.366

6.3.4. Comparación de Métodos

La evaluación estructural de las ontologías nos ofrece un conjunto de medidas absolutas, que si bien nos permiten hacernos una idea de los resultados obtenidos, no son realmente adecuadas para la comparación de métodos. No tiene sentido evaluar los resultados en términos de número de nodos o de profundidad, ya que estas medidas no son indicativo de la semántica codificada en la estructura.

Para comprobar si hay algún método que destaque de forma clara sobre el resto, realizamos un diagrama de cajas sobre los valores de cobertura (*recall*), tal y como se muestra en la Figura 6.1. Como podemos observar, el grado de solapamiento es muy alto y por tanto no podemos determinar de forma clara que uno de los métodos sea mejor que los demás.

Debido al alto grado de solapamiento, utilizaremos el test estadístico no paramétrico de Friedman [Friedman, 1937, Friedman, 1940] para encontrar diferencias significativas entre los distintos métodos. La entrada a este este test es una matriz en la que las columnas representan los métodos a evaluar y las filas los conjuntos de datos sobre los que se han evaluado los métodos. Como hemos comentado anteriormente, evaluaremos cada método por cada conjunto de datos, indicando el valor obtenido para la cobertura. Este test produce un ranking ordenado con los mejores algoritmos en orden creciente. Aquellos algoritmos con valores bajos de ranking serán mejores que otros con valores mas altos. Con estos valores de ranking se determina el mejor método y se compara con el resto de métodos. En caso de detectarse diferencias significativas, realizaremos un test *post hoc* para determinar si la diferencia existente entre métodos es significativa para un valor de $\alpha = 0.05$. Emplearemos el test de Hommel [Hommel, 1988] para las comprobaciones *post hoc*.

Aplicaremos el test por bloques, cada bloque estará compuesto por evaluaciones realizadas para un mismo valor de umbral. Comenzaremos realizando los test para 0.6 e iremos incrementando el umbral hasta llegar a 1. Como hemos comentado anteriormente, para valores de umbral más altos existen mayores restricciones y por tanto el comportamiento de los métodos tiende a ser más homogéneo. El caso límite es, cuando el umbral es 1 por lo que estaríamos en el caso base, y el comportamiento sería prácticamente igual (aún existe la posibilidad de que se acepte alguna categoría con un valor de comparación 1, lo cual mejoraría el caso base).

Por tanto, conforme aumentamos el umbral la variabilidad va decreciendo, esto implica que si un test estadístico no encuentra diferencias significativas a un nivel determinado de umbral, no las encontrará a niveles de umbral superiores donde la variabilidad es menor.

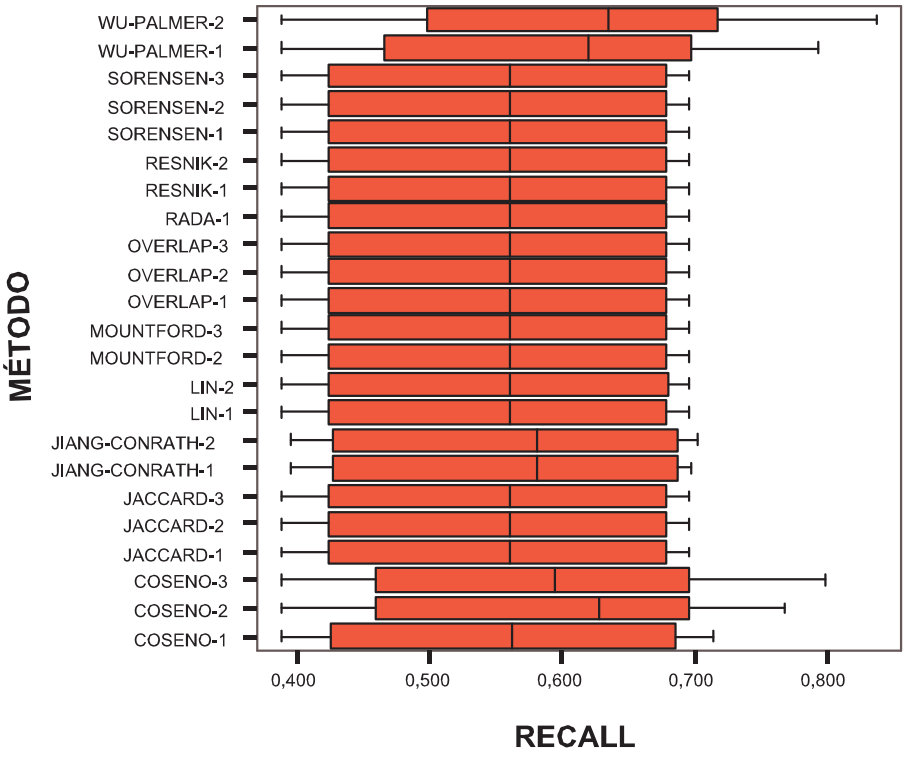


Figura 6.1: Diagrama de cajas para la cobertura de los diferentes métodos

6.4. Resultados

En esta sección analizaremos los diferentes métodos presentados. Para ello, primero analizaremos sus valores de relevancia semántica con respecto al conjunto de datos WS-209, posteriormente mostraremos los valores obtenidos por la evaluación estructural y los valores de *recall* o cobertura sobre el conjunto de datos inicial. Finalmente, realizaremos una comparación de los diferentes métodos analizados, teniendo en cuenta el valor de cobertura obtenido por el método.

Dado que podemos emplear diferentes valores de umbral, y que cada uno de estos valores proporciona diferentes resultados, vamos a emplear valores en un rango $[0.6, 1]$. Elegimos 0.6 como límite inferior, porque consideramos que un valor menor permitiría la inclusión de categorías poco relacionadas.

6.4.1. Métodos basados en Exploración de Grafos

Este conjunto de métodos presentados en la sección 5.2.1, se basan únicamente en criterios topológicos y de exploración. Desde el punto de vista semántico, resulta muy complicado determinar el resultado esperado que nos ofrecerán estos métodos, ya que no dependen en la comparación de categorías, sino únicamente en el establecimiento de unas pautas para recorrer el grafo. Es por estas razones que descartaremos el uso de estos métodos, pero aún así, consideramos que es interesante ofrecer algunos datos acerca de los resultados que proporcionan.

Relevancia Semántica

Para este tipo de métodos no es posible evaluar la relevancia semántica, puesto que están específicamente diseñados para recorrer el grafo en orden ascendente hacia la raíz a partir de una categoría base. Aunque en este apartado no podamos obtener evaluación, realmente esto no representa un problema, ya que estos métodos se emplearán junto con métodos basados en medidas que se presentan en otras secciones, y sobre estos sí que tenemos información sobre su relevancia semántica.

Evaluación Estructural

En las Tablas 6.4 y 6.5 mostramos los datos obtenidos para los diferentes métodos y datasets en la evaluación estructural. Hemos de destacar que en el caso concreto que nos ocupa, las distintas variantes del algoritmo básico resumido con longitud de camino generan la misma ontología, esto es debido a que los caminos existentes entre los nodos y su *lcs* son de tamaño 2 o menor, con lo que en todas las variantes se seleccionan los mismos nodos. Aunque en esta ocasión las variantes de los métodos no han sido de utilidad, en otras circunstancias en las que los caminos entre los nodos y su *lcs* sean de mayor longitud, el resultado variará. Dada esta circunstancia, a partir de ahora trataremos el método básico resumido con longitud de camino, como un único método por lo que podemos ignorar el resto de variantes.

Tabla 6.4: Resultados de la evaluación estructural para los métodos basados en exploración de grafos

Medida	CCIA-500	MEDLINE-500	SPORT-500	MISC-500
Exploración Básico				
Número de nodos	2847	2991	4133	3006
Número máximo de padres	10	10	82	10
Número medio de padres	3	3	3	3
Moda del número de padres	2	2	2	2
Número máximo de hermanos	26	30	83	30
Número medio de hermanos	3	3	3	3
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	31	38	39	31
Profundidad máxima	389	372	421	407
Anchura máxima	2188	2442	3571	2346
Anchura media	626	877	1652	662
Complejidad	0.71	0.73	0.78	0.72
Ratio de dispersión en hojas	0.01	0.01	0.01	0.01
Dispersión máxima en hojas	2	2	2	2
Ratio de dispersión en hermanos	2.18	2.22	2.38	2.19
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite
Básico Resumido				
Número de nodos	31	50	28	30
Número máximo de padres	4	4	2	4
Número medio de padres	2	2	2	2
Moda del número de padres	1	1	1	1
Número máximo de hermanos	8	14	4	8
Número medio de hermanos	4	4	3	4
Moda del número de hermanos	1	2	1	2
Número de nodos hoja	24	36	22	23
Profundidad máxima	3	2	2	3
Profundidad media	1.92	1.81	1.54	1.89
Anchura máxima	17	27	17	16
Anchura media	11	25	14	10
Complejidad	6.2	4.17	9.33	5
Ratio de dispersión en hojas	0.77	0.72	0.79	0.77
Dispersión máxima en hojas	6	14	4	7
Ratio de dispersión en hermanos	0.87	1.12	0.46	0.9
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite
Longitud de Camino (N=3)				
Número de nodos	184	246	200	192
Número máximo de padres	8	7	11	7
Número medio de padres	2	2	2	2
Moda del número de padres	1	1	1	1
Número máximo de hermanos	5	7	5	6
Número medio de hermanos	2	2	2	2
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	19	31	22	21
Profundidad máxima	13	11	8	11
Profundidad media	6.74	5.67	3.63	5.15
Anchura máxima	85	101	98	83
Anchura media	33	42	32	30
Complejidad	4.49	5.47	7.69	6.62
Ratio de dispersión en hojas	0.1	0.13	0.11	0.11
Dispersión máxima en hojas	2	2	2	2
Ratio de dispersión en hermanos	1.28	1.21	1.08	1.19
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite

Tabla 6.5: Resultados de la evaluación estructural para los métodos basados en exploración de grafos (cont.)

Medida	CCIA-500	MEDLINE-500	SPORT-500	MISC-500
Básico Resumido con Longitud de Camino (a-b-c-d N=2)				
Número de nodos	53	82	39	58
Número máximo de padres	3	3	2	3
Número medio de padres	2	2	2	2
Moda del número de padres	1	1	1	1
Número máximo de hermanos	9	10	4	8
Número medio de hermanos	2	2	2	2
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	25	39	23	25
Profundidad máxima	6	7	4	7
Profundidad media	3.16	2.96	1.96	3.37
Anchura máxima	18	29	17	18
Anchura media	10	15	10	10
Complejidad	0.15	0.22	0.05	0.21
Ratio de dispersión en hojas	0.47	0.48	0.59	0.43
Dispersión máxima en hojas	2	3	2	3
Ratio de dispersión en hermanos	0.89	0.96	0.62	0.97
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite
Básico de Camino Mínimo				
Número de nodos	53	85	56	52
Número máximo de padres	1	1	1	1
Número medio de padres	1	1	1	1
Moda del número de padres	1	1	1	1
Número máximo de hermanos	3	2	2	4
Número medio de hermanos	2	2	2	2
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	25	33	22	24
Profundidad máxima	4	5	5	4
Profundidad media	2.2	2.61	2.59	2.29
Anchura máxima	23	32	21	21
Anchura media	14	17	12	13
Complejidad	53	85	56	52
Ratio de dispersión en hojas	0.47	0.39	0.39	0.46
Dispersión máxima en hojas	1	1	1	2
Ratio de dispersión en hermanos	0.57	0.62	0.62	0.6
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite
Taxonómico Combinado (N=2)				
Número de nodos	157	254	176	177
Número máximo de padres	5	7	12	7
Número medio de padres	2	2	2	2
Moda del número de padres	1	1	1	1
Número máximo de hermanos	11	15	30	13
Número medio de hermanos	2	2	2	2
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	23	35	22	22
Profundidad máxima	8	7	8	7
Profundidad media	5.32	5.32	4.82	5.34
Anchura máxima	48	72	55	51
Anchura media	24	41	24	29
Complejidad	6.04	7.06	12.57	6.32
Ratio de dispersión en hojas	0.15	0.14	0.12	0.12
Dispersión máxima en hojas	3	3	2	2
Ratio de dispersión en hermanos	1.25	1.23	1.18	1.2
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite

6.4.2. Cobertura del conjunto de datos inicial

La cobertura del conjunto de datos a partir del cual se ha generado la ontología para los métodos basados en exploración del grafo se presenta en la Tabla 6.6.

Tabla 6.6: Resultados de la cobertura sobre el conjunto original para los métodos basados en exploración de grafos

Método	CCIA-500	MEDLINE-500	SPORT-500	MISC-500
Exploración Básico	**0.98**	**1**	**0.898**	1
Básico Resumido	0.696	0.724	0.358	0.366
Longitud de Camino	0.876	0.782	0.59	0.66
Básico Resumido con Longitud de Camino (a)	0.696	0.68	0.464	0.428
Básico Resumido con Longitud de Camino (b)	0.696	0.68	0.464	0.428
Básico Resumido con Longitud de Camino (c)	0.696	0.68	0.464	0.428
Básico de Camino Mínimo	0.518	0.578	0.296	0.378
Taxonómico Combinado	0.794	0.8	0.442	0.5

Comparación de Métodos

La comparación de métodos exploratorios resulta complicada, debido a que cada uno tiene distintos parámetros con semánticas diferenciadas. El parámetro N en el método de longitud de camino, no significa lo mismo que en el método básico resumido con longitud de camino. Ante la dificultad de establecer una comparación en condiciones de igualdad para los métodos, y debido a la imposibilidad de realizar un tratamiento semántico de la información, a partir de ahora descartaremos los métodos exploratorios en el análisis de los resultados.

6.4.3. Métodos basados en Índices Estadísticos

Como ya comentamos en la sección 5.2.3, el principal inconveniente de estos métodos es que proporcionan valores de similitud muy bajos. Aunque los valores están normalizados en el intervalo $[0, 1]$, los valores de similitud son tan bajos que no es posible establecer de forma intuitiva una correspondencia con un rango de valores que sea manejable por un humano. Dado que para establecer una comparación entre métodos debemos establecer un valor de umbral, valores normales como 0.6 son excesivamente altos para estos métodos, por lo que van a proporcionar valores equivalentes al caso base. Aunque esta apreciación se puede ver con los datos obtenidos, vamos a establecer una comparación más detallada. En cualquier caso, trataremos de determinar el mejor de entre los métodos basados en índices estadísticos.

Relevancia Semántica

Los valores de correlación respecto al conjunto WS-209 se presentan en la tabla 6.7

Tabla 6.7: Correlación de los métodos basados en índices estadísticos

Método	Correlación WS-209
Media Jaccard	0.2559
Mejor Jaccard	0.2993
Media Sorensen	0.2741
Mejor Sorensen	0.3122
Media Mountford	0.2303
Mejor Mountford	0.2841
Media Overlap	0.2776
Mejor Overlap	**0.3296**

Podemos observar que el mejor valor lo obtiene el método de *Overlap*, pero todos los valores obtenidos para los diferentes métodos son bastante similares.

Evaluación Estructural

En la Tabla 6.8 se muestran los resultados de la evaluación estructural para los métodos estadísticos. Hemos fijado el valor de umbral a 0.6, el menos restrictivo, y que por tanto mayor variabilidad puede aportar a los resultados. Como se puede observar en las distintas tablas, la evaluación para este umbral es equivalente al caso base para todos los métodos. Esto quiere decir que para valores en el intervalo $[0.6, 1]$ los resultados siempre serán iguales al caso base.

6.4.4. Cobertura del conjunto de datos inicial

La cobertura del conjunto de datos a partir del cual se ha generado la ontología para los métodos basados en exploración del grafo se presenta en la Tabla 6.9. Como en el caso anterior de la evaluación estructural, vemos que nos encontramos con el caso base, y que por lo tanto todos los métodos se comportan de igual forma para valores de umbral dentro del rango que estamos utilizando.

Comparación de Métodos

Como hemos comprobado en las evaluaciones anteriores, los métodos tienen el mismo comportamiento. Aunque cada uno ofrece medidas diferentes, los valores de similitud ofrecidos son tan bajos que quedan por debajo del rango en el que estamos trabajando. A pesar de tener evidencias más que suficientes para determinar la igualdad de los métodos empleados, incluso cambios en el ámbito de comparación no hacen que los resultados mejoren, vamos a realizar un análisis estadístico.

Vamos a realizar el estudio de los métodos para valores de umbral 0.6. Los resultados obtenidos serán extrapolables a todos los valores de umbral del intervalo $[0.6, 1]$, puesto que el aumento en el umbral produce resultados más restrictivos, y ya nos encontramos en el caso base.

Tabla 6.8: Resultados de la evaluación estructural para los métodos basados en índices estadísticos

Medida	CCIA-500	MEDLINE-500	SPORT-500	MISC-500
Jaccard, Sorensen-Dice, Mountford, Overlap (Inicial t=0.6)				
Número de nodos	26	35	22	24
Número máximo de padres	1	1	1	1
Número medio de padres	1	1	1	1
Moda del número de padres	1	1	1	1
Número máximo de hermanos	0	0	0	0
Número medio de hermanos	0	0	0	0
Moda del número de hermanos	0	0	0	0
Número de nodos hoja	26	35	22	24
Profundidad máxima	1	1	1	1
Profundidad media	1	1	1	1
Anchura máxima	26	35	22	24
Anchura media	26	35	22	24
Complejidad	0	0	0	0
Ratio de dispersión en hojas	1	1	1	1
Dispersión máxima en hojas	0	0	0	0
Ratio de dispersión en hermanos	0	0	0	0
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite
Jaccard, Sorensen-Dice, Mountford, Overlap (Actual t=0.6)				
Número de nodos	26	35	22	24
Número máximo de padres	1	1	1	1
Número medio de padres	1	1	1	1
Moda del número de padres	1	1	1	1
Número máximo de hermanos	0	0	0	0
Número medio de hermanos	0	0	0	0
Moda del número de hermanos	0	0	0	0
Número de nodos hoja	26	35	22	24
Profundidad máxima	1	1	1	1
Profundidad media	1	1	1	1
Anchura máxima	26	35	22	24
Anchura media	26	35	22	24
Complejidad	0	0	0	0
Ratio de dispersión en hojas	1	1	1	1
Dispersión máxima en hojas	0	0	0	0
Ratio de dispersión en hermanos	0	0	0	0
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite
Jaccard, Sorensen-Dice, Mountford, Overlap (Acumulado t=0.6)				
Número de nodos	26	35	22	24
Número máximo de padres	1	1	1	1
Número medio de padres	1	1	1	1
Moda del número de padres	1	1	1	1
Número máximo de hermanos	0	0	0	0
Número medio de hermanos	0	0	0	0
Moda del número de hermanos	0	0	0	0
Número de nodos hoja	26	35	22	24
Profundidad máxima	1	1	1	1
Profundidad media	1	1	1	1
Anchura máxima	26	35	22	24
Anchura media	26	35	22	24
Complejidad	0	0	0	0
Ratio de dispersión en hojas	1	1	1	1
Dispersión máxima en hojas	0	0	0	0
Ratio de dispersión en hermanos	0	0	0	0
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite

Tabla 6.9: Resultados de la cobertura sobre el conjunto original para los métodos basados en índices estadísticos

Método	**CCIA-500**	**MEDLINE-500**	**SPORT-500**	**MISC-500**
Categoría Inicial (t=0.6)				
Jaccard	0.696	0.662	0.46	0.388
Sorensen-Dice	0.696	0.662	0.46	0.388
Mountford	0.696	0.662	0.46	0.388
Overlap	0.696	0.662	0.46	0.388
Categoría Actual (t=0.6)				
Jaccard	0.696	0.662	0.46	0.388
Sorensen-Dice	0.696	0.662	0.46	0.388
Mountford	0.696	0.662	0.46	0.388
Overlap	0.696	0.662	0.46	0.388
Categoría Acumulada (t=0.6)				
Jaccard	0.696	0.662	0.46	0.388
Sorensen-Dice	0.696	0.662	0.46	0.388
Mountford	0.696	0.662	0.46	0.388
Overlap	0.696	0.662	0.46	0.388

La tabla 6.10 muestra los valores de ranking obtenidos por el test de Friedman. Se obtuvo un *p-valor* de 1 en esta comparación. Es decir, no se encontraron diferencias entre los métodos. Cada método está seguido de un número, este número indica que estamos utilizando el ámbito de comparación con la categoría actual (1), con la categoría actual (2) o con la categoría acumulada (3). Como se puede observar, se ha determinado que todos los métodos son iguales, por el *p-valor* obtenido y porque tienen el mismo valor de ranking. Por tanto, no sería necesario realizar una prueba *post hoc*. Sin embargo la haremos para confirmar los resultados, para ello realizamos un test *post hoc* usando Hommel.

Tabla 6.10: Ranking medio de los métodos (Friedman)

Método	Ranking
JACCARD-1	6.5
JACCARD-2	6.5
JACCARD-3	6.5
SORENSEN-1	6.5
SORENSEN-2	6.5
SORENSEN-3	6.5
MOUNTFORD-1	6.5
MOUNTFORD-2	6.5
MOUNTFORD-3	6.5
OVERLAP-1	6.5
OVERLAP-2	6.5
OVERLAP-3	6.5

Los resultados para el test *post hoc* que se muestran en la tabla 6.11, ponen de manifiesto que, efectivamente, las diferencias entre los distintos métodos no son significativas. En realidad, tal y como ya habíamos podido observar por los datos obtenidos, todos los métodos basados en índices estadísticos producen los mismos resultados. Además, si tene-

mos en cuenta que los resultados coinciden con el caso base, tenemos suficientes elementos de juicio como para descartar el uso de métodos empleando métodos estadísticos.

Tabla 6.11: p-valores ajustados (Friedman)

i	Método	p sin ajustar	p_{Hommel}
1	JACCARD-1 vs. JACCARD-2	1	1
2	JACCARD-1 vs. JACCARD-3	1	1
3	JACCARD-1 vs. SORENSEN-1	1	1
4	JACCARD-1 vs. SORENSEN-2	1	1
5	JACCARD-1 vs. SORENSEN-3	1	1
6	JACCARD-1 vs. MOUNTFORD-1	1	1
7	JACCARD-1 vs. MOUNTFORD-2	1	1
8	JACCARD-1 vs. MOUNTFORD-3	1	1
9	JACCARD-1 vs. OVERLAP-1	1	1
10	JACCARD-1 vs. OVERLAP-2	1	1
11	JACCARD-1 vs. OVERLAP-3	1	1

6.4.5. Métodos basados en Similitud Semántica

Este conjunto de métodos fue presentado en la sección 5.2.4. Como ya se comentó entonces, estas medidas basadas únicamente en criterios topológicos no parecen ser muy discriminantes. Sin embargo, aunque no se tenga en cuenta el contenido de las categorías, la información inherente a la estructura del grafo puede ser explotada de forma satisfactoria.

Relevancia Semántica

Los valores de correlación obtenidos para el conjunto de medidas basadas en similitud semántica se muestra en la Tabla 6.12. Como se puede observar el método con mejor correlación es Leacock-Chodorow.

Tabla 6.12: Correlación de los métodos de similitud semántica

Método	**Correlación WS-209**
Media Rada	-0.2974
Mejor Rada	-0.2942
Media Leacock-Chodorow	0.3358
Mejor Leacock-Chodorow	**0.3635**
Media Wu-Palmer	0.2764
Mejor Wu-Palmer	0.2635

Evaluación Estructural

En el caso de las medidas de similitud semántica que ahora nos ocupan, los valores de similitud se mueven en distintos rangos, para Rada y Wu-Palmer los valores obtenidos se mueven aproximadamente en el intervalo $[0, 1]$ (en el caso de Rada particularmente tiende

a 0, pero no llega a alcanzar dicho valor), sin embargo la media de Leacock-Chodorow no está normalizada. Esto supone un gran problema ya que al establecer los valores de umbral no estamos realizando un evaluación uniforme. Esto, unido al hecho de que es realmente difícil determinar un valor de umbral para esta medida de forma intuitiva, nos llevan a descartarla. De este modo sólo mostraremos los valores obtenidos para Rada y Wu-Palmer.

En la Tabla 6.13 podemos observar como para la medida de Rada, el valor para un umbral de 0.6 coincide con el caso base, esto se debe a la forma en que se calcula esta medida. Según Rada, la similitud entre dos nodos padre e hijo es de 0.5, por tanto, todos los valores de umbral superiores nos proporcionan el caso base. Vemos por tanto, que esta medida no opera en el rango de umbrales que estamos manejando.

La medida de Rada no tiene sentido aplicarla con el uso de la categoría actual, ya que al actualizar el contexto de comparación y al basarse esta medida en la distancia entre nodos, dicha distancia será constante y por tanto, no discriminante. Si utilizásemos la comparación con la categoría actual y un umbral de 0.5 o menor, seleccionaríamos todo el grafo de categorías.

El operar fuera de rango y la imposibilidad de aplicar la medida con otros ámbitos de comparación hacen que podamos descartar el uso de Rada como medida para la selección de categorías.

Tabla 6.13: Resultados de la evaluación estructural para el método de Rada

Medida	**CCIA-500**	**MEDLINE-500**	**SPORT-500**	**MISC-500**
Rada (Inicial t=0.6)				
Número de nodos	26	35	22	24
Número máximo de padres	1	1	1	1
Número medio de padres	1	1	1	1
Moda del número de padres	1	1	1	1
Número máximo de hermanos	0	0	0	0
Número medio de hermanos	0	0	0	0
Moda del número de hermanos	0	0	0	0
Número de nodos hoja	26	35	22	24
Profundidad máxima	1	1	1	1
Profundidad media	1	1	1	1
Anchura máxima	26	35	22	24
Anchura media	26	35	22	24
Complejidad	0	0	0	0
Ratio de dispersión en hojas	1	1	1	1
Dispersión máxima en hojas	0	0	0	0
Ratio de dispersión en hermanos	0	0	0	0
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite

En la Tabla 6.14 podemos ver los resultados obtenidos para el método de Wu-Palmer usando los ámbitos de comparación de la categoría inicial y la categoría actual. Vemos que el número de nodos es bastante elevado con respecto al caso base.

Tabla 6.14: Resultados de la evaluación estructural para los métodos basados en similitud semántica

Medida	**CCIA-500**	**MEDLINE-500**	**SPORT-500**	**MISC-500**
Wu-Palmer (Inicial t=0.6)				
Número de nodos	87	118	64	80
Número máximo de padres	3	3	2	2
Número medio de padres	2	2	2	2
Moda del número de padres	1	1	1	1
Número máximo de hermanos	5	3	2	4
Número medio de hermanos	2	2	2	2
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	23	36	23	22
Profundidad máxima	15	10	6	15
Profundidad media	5.83	4.34	2.88	5.08
Anchura máxima	26	30	22	25
Anchura media	11	15	12	9
Complejidad	0.2	0.08	0.02	0.2
Ratio de dispersión en hojas	0.26	0.31	0.36	0.28
Dispersión máxima en hojas	2	2	1	2
Ratio de dispersión en hermanos	0.98	0.86	0.67	0.94
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite
Wu-Palmer (Actual t=0.6)				
Número de nodos	159	201	107	157
Número máximo de padres	4	4	5	4
Número medio de padres	2	2	2	2
Moda del número de padres	1	1	1	1
Número máximo de hermanos	6	6	3	6
Número medio de hermanos	2	2	2	2
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	22	36	24	20
Profundidad máxima	44	41	19	40
Profundidad media	26.04	19.92	5.75	22.39
Anchura máxima	40	38	23	30
Anchura media	25	20	8	20
Complejidad	0.19	0.14	0.07	0.18
Ratio de dispersión en hojas	0.14	0.18	0.22	0.13
Dispersión máxima en hojas	2	2	1	2
Ratio de dispersión en hermanos	1.14	1.06	0.89	1.12
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite

6.4.6. Cobertura del conjunto de datos inicial

La cobertura del conjunto de datos a partir del cual se ha generado la ontología para los métodos basados en exploración del grafo se presenta en la Tabla 6.15. Recordemos que no vamos a presentar resultados para Leacock-Chodorow por no estar normalizada, y por tanto, no se puede establecer una comparación directa con el resto de métodos. En los resultados podemos ver cómo claramente la medida de Wu-Palmer obtiene una mayor cobertura, efecto que sin duda se debe al alto número de categorías seleccionadas.

Tabla 6.15: Resultados de la cobertura sobre el conjunto original para los métodos basados en exploración de grafos

Método	**CCIA-500**	**MEDLINE-500**	**SPORT-500**	**MISC-500**
Categoría Inicial				
Rada	0.696	0.662	0.46	0.388
Wu-Palmer	0.794	0.712	0.498	0.578
Categoría Actual				
Wu-Palmer	**0.808**	**0.838**	**0.518**	**0.608**

Comparación de Métodos

Para las medidas de similitud semántica utilizaremos el test de Friedman para evaluar los métodos de Rada comparando con la categoría inicial (1) (la única opción viable) y Wu-Palmer empleando la comparación con la categoría inicial (1), y con la categoría actual (2), utilizando un valor de umbral 0.6. El test de Friedman obtiene un *p-valor* de 0.018316, lo cual indica que hay diferencias significativas. En la Tabla 6.16 podemos ver como se ha determinado que Wu-Palmer es el mejor método, concretamente su variante de comparación con la categoría actual, seguido de Wu-Palmer comparando con la categoría inicial y después Rada. Tomando el mejor método, comprobaremos si las diferencias encontradas con el resto de métodos son estadísticamente significativas empleando el test *post hoc* de Hommel, tal y como se muestra en la Tabla 6.17. Podemos ver como efectivamente, la diferencia entre Wu-Palmer y Rada es significativa, sin embargo no se puede determinar que la diferencia entre las dos variantes de Wu-Palmer lo sea.

Tabla 6.16: Ranking de los métodos de similitud semántica (Friedman)

Algoritmo	Ranking
RADA-1	3
WU-PALMER-1	2
WU-PALMER-2	1

Tabla 6.17: p-valores ajustados (Friedman)

i	algoritmo	p sin ajustar	p_{Hommel}
1	WU-PALMER-2 vs. RADA-1	0.004678	0.009355
2	WU-PALMER-2 vs. WU-PALMER-1	0.157299	0.157299

6.4.7. Métodos basados en Contenido de la Información

En la sección 5.2.5 presentamos los métodos que usan medidas de contenido de información. Ya comentamos en dicha sección, que el principal inconveniente de estas medidas es la necesidad de adaptar el grafo de categorías, y la pérdida de información que dicha adaptación conlleva. Veamos el comportamiento de estos métodos.

Relevancia Semántica

La Tabla 6.18 muestra los valores de correlación para los métodos de contenido de la información. Podemos observar que el método de Lin es el que emite juicios de relación mas cercanos a aquellos que realizaría un humano.

Tabla 6.18: Correlación de los métodos de contenido de la información

Método	Correlación WS-209
Media Resnik	0.2670
Mejor Resnik	0.3032
Media Lin	0.2760
Mejor Lin	**0.3113**
Media Jiang-Conrath	-0.0565
Mejor Jiang-Conrath	0.1079

Evaluación Estructural

En la Tabla 6.19 podemos ver los valores obtenidos para los métodos de contenido de información usando un umbral de 0.6, y usando comparación con la categoría inicial. Podemos ver que todos los métodos mejoran el caso base, y que el método de Jiang-Conrath es el que más categorías ha seleccionado.

Cuando se cambia el ámbito de comparación a la categoría actual, tal y como se muestra en la Tabla 6.20, nos encontramos con que los valores son muy similares. Para Resnik coinciden con el caso anterior, mientras que para Lin y Jiang-Conrath aumentan ligeramente.

6.4.8. Cobertura del conjunto de datos inicial

La cobertura del conjunto de datos a partir del cual se ha generado la ontología, para los métodos basados en exploración del grafo, se presenta en la Tabla 6.21. Como se puede

Tabla 6.19: Resultados de la evaluación estructural para los métodos basados en contenido de la información

Medida	**CCIA-500**	**MEDLINE-500**	**SPORT-500**	**MISC-500**
Resnik (Inicial t=0.6)				
Número de nodos	27	42	29	27
Número máximo de padres	1	1	1	1
Número medio de padres	1	1	1	1
Moda del número de padres	1	1	1	1
Número máximo de hermanos	1	1	1	1
Número medio de hermanos	1	1	1	1
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	26	35	22	24
Profundidad máxima	2	3	4	3
Profundidad media	1.04	1.2	1.32	1.12
Anchura máxima	26	35	22	24
Anchura media	14	14	8	9
Complejidad	0	0	0	0
Ratio de dispersión en hojas	0.96	0.83	0.76	0.89
Dispersión máxima en hojas	1	1	1	1
Ratio de dispersión en hermanos	0.04	0.17	0.24	0.11
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite
Lin (Inicial t=0.6)				
Número de nodos	31	49	36	32
Número máximo de padres	1	1	1	1
Número medio de padres	1	1	1	1
Moda del número de padres	1	1	1	1
Número máximo de hermanos	1	2	1	1
Número medio de hermanos	1	2	1	1
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	26	35	22	24
Profundidad máxima	2	4	5	4
Profundidad media	1.19	1.46	1.64	1.33
Anchura máxima	26	33	22	24
Anchura media	16	13	8	8
Complejidad	0	0	0	0
Ratio de dispersión en hojas	0.84	0.71	0.61	0.75
Dispersión máxima en hojas	1	2	1	1
Ratio de dispersión en hermanos	0.16	0.33	0.39	0.25
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite
Jiang-Conrath (Inicial t=0.6)				
Número de nodos	46	77	52	42
Número máximo de padres	3	2	4	2
Número medio de padres	2	2	2	2
Moda del número de padres	1	1	1	1
Número máximo de hermanos	2	2	4	2
Número medio de hermanos	2	2	2	2
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	26	35	22	23
Profundidad máxima	5	7	5	7
Profundidad media	1.75	2.44	2.32	2.04
Anchura máxima	27	33	22	23
Anchura media	10	13	11	7
Complejidad	0.02	0.05	0.02	0.05
Ratio de dispersión en hojas	0.57	0.45	0.42	0.55
Dispersión máxima en hojas	1	2	1	1
Ratio de dispersión en hermanos	0.46	0.62	0.63	0.5
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite

Tabla 6.20: Resultados de la evaluación estructural para los métodos basados en contenido de la información

Medida	CCIA-500	MEDLINE-500	SPORT-500	MISC-500
Resnik (Actual t=0.6)				
Número de nodos	27	42	29	27
Número máximo de padres	1	1	1	1
Número medio de padres	1	1	1	1
Moda del número de padres	1	1	1	1
Número máximo de hermanos	1	1	1	1
Número medio de hermanos	1	1	1	1
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	26	35	22	24
Profundidad máxima	2	3	4	3
Profundidad media	1.04	1.2	1.32	1.12
Anchura máxima	26	35	22	24
Anchura media	14	14	8	9
Complejidad	0	0	0	0
Ratio de dispersión en hojas	0.96	0.83	0.76	0.89
Dispersión máxima en hojas	1	1	1	1
Ratio de dispersión en hermanos	0.04	0.17	0.24	0.11
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite
Lin (Actual t=0.6)				
Número de nodos	31	50	36	32
Número máximo de padres	1	1	1	1
Número medio de padres	1	1	1	1
Moda del número de padres	1	1	1	1
Número máximo de hermanos	1	2	1	1
Número medio de hermanos	1	2	1	1
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	26	35	22	24
Profundidad máxima	2	4	5	4
Profundidad media	1.19	1.49	1.64	1.33
Anchura máxima	26	33	22	24
Anchura media	16	13	8	8
Complejidad	0	0	0	0
Ratio de dispersión en hojas	0.84	0.7	0.61	0.75
Dispersión máxima en hojas	1	2	1	1
Ratio de dispersión en hermanos	0.16	0.34	0.39	0.25
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite
Jiang-Conrath (Actual t=0.6)				
Número de nodos	48	76	56	41
Número máximo de padres	3	2	4	2
Número medio de padres	2	2	2	2
Moda del número de padres	1	1	1	1
Número máximo de hermanos	2	2	5	2
Número medio de hermanos	2	2	2	2
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	26	35	22	23
Profundidad máxima	5	7	8	7
Profundidad media	1.83	2.41	2.96	2
Anchura máxima	28	33	21	23
Anchura media	11	13	8	7
Complejidad	0.04	0.05	0.05	0.05
Ratio de dispersión en hojas	0.54	0.46	0.39	0.56
Dispersión máxima en hojas	1	2	1	1
Ratio de dispersión en hermanos	0.48	0.62	0.73	0.49
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite

apreciar Resnik y Lin, no consiguen mejorar la cobertura del caso base en ninguno de los ámbitos de comparación, a pesar de que habían incluido categorías adicionales. Sin embargo, para Jiang-Conrath se aprecia una leve mejora de la cobertura y se puede ver que el método de comparación con la categoría actual es ligeramente superior.

Tabla 6.21: Resultados de la cobertura sobre el conjunto original para los métodos basados en IC

Método	**CCIA-500**	**MEDLINE-500**	**SPORT-500**	**MISC-500**
Categoría Inicial (t=0.5)				
Resnik	0.696	0.662	0.46	0.388
Lin	0.696	0.662	0.46	0.388
Jiang-Conrath	0.698	**0.678**	0.494	**0.396**
Categoría Actual (t=0.5)				
Resnik	0.696	0.662	0.46	0.388
Lin	0.696	0.664	0.46	0.388
Jiang-Conrath	**0.702**	**0.678**	**0.496**	**0.396**

Comparación de Métodos

Mediante el test de Friedman para un valor de umbral 0.6 vamos a determinar cual de los métodos basados en contenido de la información es mejor. Para ambos métodos utilizaremos la comparación con la categoría inicial (1) y la actual (2). El ranking obtenido para un *p-valor* de 0.015162, se muestra en la Tabla 6.22. Dado que se han detectado diferencias significativas, aplicamos en el test de Hommel. Según los datos observados en la tabla 6.23 podemos decir que el método de Jiang-Conrath en su variante de comparación con la categoría actual presenta diferencias estadísticas significativas con los métodos de Resnik y Lin usando comparación con la categoría inicial. Sin embargo no podemos determinar que existan diferencias significativas entre Jiang-Conrath en sus dos variantes y Lin comparando con la categoría actual.

Tabla 6.22: Ranking de los métodos de contenido de información (Friedman)

Algoritmo	Ranking
RESNIK-1	4.625
RESNIK-2	4.625
LIN-1	4.625
LIN-2	4.125
JIANG-CONRATH-1	1.75
JIANG-CONRATH-2	1.25

6.4.9. Métodos basados en Recuperación de Información

En la sección 5.2.6 se presentaron los métodos basados en recuperación de información, consistentes en la comparación del contenido de las categorías utilizando el texto contenido

Tabla 6.23: p-valores ajustados (Friedman)

i	algoritmo	p sin ajustar	p_{Hommel}
1	JIANG-CONRATH-2 vs. RESNIK-1	0.010733	0.0322
2	JIANG-CONRATH-2 vs. RESNIK-2	0.010733	0.0322
3	JIANG-CONRATH-2 vs. LIN-1	0.010733	0.0322
4	JIANG-CONRATH-2 vs. LIN-2	0.029758	0.059516
5	JIANG-CONRATH-2 vs. JIANG-CONRATH-1	0.705457	0.705457

en sus páginas. Como ya comentamos, el principal inconveniente de esta aproximación es la cantidad de procesamiento necesario para poder establecer la comparación, y los requisitos de memoria necesarios. Veamos si esta alternativa basada en el contenido puede competir con el resto, las cuales están mas orientadas a explotar las información codificada en la propia taxonomía.

Relevancia Semántica

En la Tabla 6.24 mostramos los valores de correlación obtenidos para el método. Como podemos comprobar es el valor más alto de los obtenidos hasta el momento, lo cual nos indica que el procesamiento del contenido de las categorías nos da una mejor idea de su semántica, que el uso de criterios meramente estructurales.

Tabla 6.24: Correlación de los métodos basados en recuperación de información

Método	Correlación WS-209
Media Coseno	**0.4993**
Mejor Coseno	0.4176

Evaluación Estructural

En la Tabla 6.25 podemos ver los valores obtenidos para el método del Coseno, empleando los tres ámbitos de comparación y con un umbral de 0.6. Como podemos observar se seleccionan una gran cantidad de categorías nuevas y las ontologías aumentan en complejidad.

6.4.10. Cobertura del conjunto de datos inicial

La cobertura del conjunto de datos a partir del cual se ha generado la ontología, para los métodos basados en exploración del grafo con umbral 0.6, se presenta en la Tabla 6.26. Podemos observar como todos mejoran el caso base. La mejora es especialmente notable para la comparación con la categoría actual y acumulada. Es interesante observar que para los conjuntos de datos SPORT y MISC, la comparación con la categoría actual obtiene mejores resultados y empata respectivamente, con respecto a la categoría acumulada.

Tabla 6.25: Resultados de la evaluación estructural para los métodos basados en recuperación de información

Medida	**CCIA-500**	**MEDLINE-500**	**SPORT-500**	**MISC-500**
Coseno (Inicial t=0.6)				
Número de nodos	52	103	47	64
Número máximo de padres	3	7	10	3
Número medio de padres	2	2	2	2
Moda del número de padres	1	1	1	1
Número máximo de hermanos	2	8	1	3
Número medio de hermanos	2	2	1	2
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	24	35	22	24
Profundidad máxima	9	8	3	6
Profundidad media	2.4	3.61	1.72	2.73
Anchura máxima	25	42	36	32
Anchura media	7	18	17	14
Complejidad	0.1	0.21	0.11	0.2
Ratio de dispersión en hojas	0.46	0.34	0.47	0.38
Dispersión máxima en hojas	2	2	1	1
Ratio de dispersión en hermanos	0.63	0.91	0.53	0.73
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite
Coseno (Actual t=0.6)				
Número de nodos	100	231	226	195
Número máximo de padres	4	7	10	5
Número medio de padres	2	2	2	2
Moda del número de padres	1	1	1	1
Número máximo de hermanos	4	10	6	7
Número medio de hermanos	2	2	2	2
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	21	36	25	24
Profundidad máxima	11	22	19	18
Profundidad media	4.92	12.23	10.7	8.71
Anchura máxima	37	80	67	62
Anchura media	16	41	35	27
Complejidad	0.27	0.35	0.3	0.35
Ratio de dispersión en hojas	0.21	0.16	0.11	0.12
Dispersión máxima en hojas	2	2	1	2
Ratio de dispersión en hermanos	0.96	1.29	1.21	1.25
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite
Coseno (Acumulada t=0.6)				
Número de nodos	154	220	125	156
Número máximo de padres	6	7	11	5
Número medio de padres	2	2	2	2
Moda del número de padres	1	1	1	1
Número máximo de hermanos	10	9	5	7
Número medio de hermanos	2	2	2	2
Moda del número de hermanos	1	1	1	1
Número de nodos hoja	22	36	23	23
Profundidad máxima	28	22	16	30
Profundidad media	16.8	11.5	7.55	15.77
Anchura máxima	77	90	39	54
Anchura media	39	39	14	28
Complejidad	0.45	0.41	0.24	0.38
Ratio de dispersión en hojas	0.14	0.16	0.18	0.15
Dispersión máxima en hojas	2	2	1	1
Ratio de dispersión en hermanos	1.48	1.35	1.12	1.27
Complejidad genérica	OWL Lite	OWL Lite	OWL Lite	OWL Lite

Tabla 6.26: Resultados de la cobertura para métodos basados en recuperación de información con umbral 0.6

Método	**CCIA-500**	**MEDLINE-500**	**SPORT-500**	**MISC-500**
Categoría Inicial				
Coseno	0.714	0.674	0.462	0.392
Categoría Actual				
Coseno	0.768	0.722	**0.594**	**0.528**
Categoría Acumulada				
Coseno	**0.798**	**0.738**	0.526	**0.528**

Comparación de Métodos

Vamos a evaluar los resultados obtenidos por los diferentes ámbitos de comparación que empleamos con el método del Coseno, inicial (1), actual (2) y acumulado (3). Usamos el test de Friedman para establecer un ranking y la prueba *post hoc* de Hommel. La Tabla 6.27 presenta el ranking de los métodos, para un *p-valor* de 0.046771. Podemos ver que el mejor método es el que emplea la categoría acumulada, sin embargo la distancia es muy poca respecto a la comparación con la categoría actual. Para poder determinar de forma clara cual es mejor en la Tabla 6.28 tenemos los valores de significación. Como se puede observar la diferencia con respecto al uso de la categoría inicial es significativa, sin embargo, no podemos decir que la diferencia entre el método usando categoría actual y usando acumulada se significativa.

Tabla 6.27: Ranking medio de los métodos RI a umbral 0.6 (Friedman)

Algoritmo	Ranking
COSENO-1	3
COSENO-2	1.625
COSENO-3	1.375

Tabla 6.28: p-valores ajustados métodos RI a umbral 0.6 (Friedman)

i	algoritmo	p sin ajustar	p_{Hommel}
1	COSENO-3 vs. COSENO-1	0.021556	0.043113
2	COSENO-3 vs. COSENO-2	0.723674	0.723674

6.5. Selección de los métodos a emplear

El primero de los criterios que hemos analizado para la selección de métodos ha sido el de la correlación con juicios humanos usando el conjunto WS-209. En la Tabla 6.29 recordamos los valores de correlación obtenidos para los diferentes métodos.

Tabla 6.29: Correlación de los distintos métodos empleados

Método	Correlación WS-209
Métodos basados en Índices Estadísticos	
Media Jaccard	0.2559
Mejor Jaccard	0.2993
Media Sorensen	0.2741
Mejor Sorensen	0.3122
Media Mountford	0.2303
Mejor Mountford	0.2841
Media Overlap	0.2776
Mejor Overlap	**0.3296**
Métodos de Similitud Semántica	
Media Rada	-0.2974
Mejor Rada	-0.2942
Media Leacock-Chodorow	0.3358
Mejor Leacock-Chodorow	**0.3635**
Media Wu-Palmer	0.2764
Mejor Wu-Palmer	0.2635
Métodos de Contenido de la Información	
Media Resnik	0.2670
Mejor Resnik	0.3032
Media Lin	0.2760
Mejor Lin	**0.3113**
Media Jiang-Conrath	-0.0565
Mejor Jiang-Conrath	0.1079
Métodos de Recuperación de Información	
Media Coseno	***0.4993****
Mejor Coseno	0.4176

Hasta ahora hemos hecho una evaluación preliminar utilizando el umbral 0.6, ya que éste es el que presenta las mayores diferencias significativas. Utilizando otros valores recogidos como el tiempo de ejecución intentaremos determinar cual es el mejor método a utilizar en cada escenario.

Dado el alto grado de solapamiento en los valores obtenidos para los diferentes métodos, utilizaremos una prueba estadística para determinar de forma adecuada cuales son los mejores algoritmos. Comenzaremos realizando el test de Friedman para obtener los rankings para los valores de umbral 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, y 1.0. Para cada valor de umbral obtendremos los p-valores con el test de Hommel, para poder conocer para cada valor de umbral cuales tienen diferencias estadísticamente significativas con respecto al seleccionado como mejor, y por tanto son claramente peores.

La Tabla 6.30 muestra los rankings obtenidos por cada valor de umbral por los diferentes métodos, junto con el *p-valor* obtenido por el test de Friedman en cada caso. Para facilitar la presentación de los datos, en la Figura 6.2 se muestra una gráfica con los valores de ranking alcanzados por cada método utilizando distintos umbrales. El mejor método es aquel que se encuentra en posiciones del ranking mas bajas. Aunque para un valor de umbral de 0.8, el test de Friedman no encuentra diferencias significativas, la aplicación del test de Hommel no permite detectarlas, pero con unos *p-valores* bastante cercanos al límite de confianza. De este modo podemos identificar dos zonas bien diferenciadas, valores de umbral menores y mayores que 0.85.

Tabla 6.30: Rankings obtenidos para distintos valores de umbral (Friedman)

Umbrales	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1
p-valor	0.00004	0.00005	0.00040	0.00045	0.05219	0.11706	0.97228	0.97228	0.99995
JACCARD-1	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
JACCARD-2	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
JACCARD-3	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
SORENSEN-1	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
SORENSEN-2	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
SORENSEN-3	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
MOUNTFORD-1	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
MOUNTFORD-2	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
MOUNTFORD-3	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
OVERLAP-1	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
OVERLAP-2	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
OVERLAP-3	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
RADA-1	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
WU-PALMER-1	3.25	2.5	2.375	1.875	3.625	3.25	13.25	13.25	13
WU-PALMER-2	1.5	1.5	1.625	1.125	1.125	1.125	13.25	13.25	13
RESNIK-1	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
RESNIK-2	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
LIN-1	16.125	16	15.75	15.75	15	14.625	13.25	13.25	13
LIN-2	14	16	15.75	15.75	15	14.625	13.25	13.25	13
JIANG-CONRATH-1	6	6	7.875	7.875	6.5	5.5	4.25	4.25	7
JIANG-CONRATH-2	5.5	5.5	4.5	4.75	6.5	5.5	4.25	4.25	7
COSENO-1	6.5	6.375	8.5	7.375	12	12	13.25	13.25	13
COSENO-2	2.875	2.75	2.75	4.25	6.375	12	13.25	13.25	13
COSENO-3	2.375	3.375	4.625	5	8.875	12	13.25	13.25	13

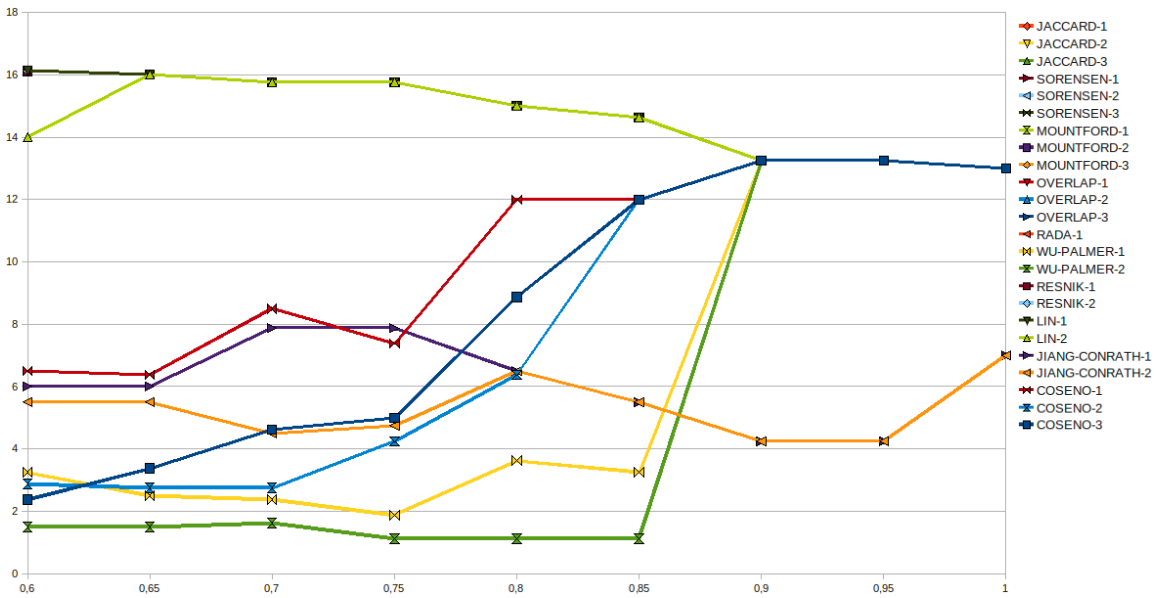


Figura 6.2: Rankings para distintos valores de umbral

Para valores de umbral mayores que 0.85, el método con mejor ranking es JIANG-CONRATH-1, sin embargo, si examinamos los resultados de las pruebas *post hoc* para dichos umbrales en la Tabla 6.31, vemos como las diferencias encontradas no son significativas. Esto significa que aunque los resultados parezcan mejores que los del resto de métodos, no se puede verificar estadísticamente. Hemos omitido los valores para el umbral 0.95 porque coinciden con los de 0.9. Así pues, si deseamos realizar el proceso de obtención de ontologías para un umbral superior a 0.85, podríamos utilizar cualquiera de los métodos puesto que estadísticamente no nos podemos decantar por ninguno en concreto. Por tanto, la selección del método a emplear debería entonces venir dada por otros criterios. Por ejemplo, se pueden tener en cuenta criterios de relevancia semántica o de tiempo de ejecución, todo dependiendo del contexto en el que estemos trabajando.

Para valores de umbral iguales o menores que 0.85, el método mejor situado es WU-PALMER-2. Al igual que en el caso anterior, verificamos esta afirmación usando el test de Friedman a distintos niveles, utilizando el método *post hoc* de Hommel. Como se puede observar en las Tablas 6.32 y 6.33 no es posible encontrar diferencias significativas entre WU-PALMER-2 y las medidas de JIANG-CONRATH y COSENO en cualquiera de sus variantes. De forma excepcional, para un umbral de 0.6 vemos que tampoco es posible encontrar diferencias significativas entre WU-PALMER-2 y LIN-2. De nuevo nos encontramos ante la imposibilidad de decantarnos por un método en concreto, así que debemos usar otros criterios como el tiempo de ejecución o la relevancia semántica.

En general, podemos decir que para valores por encima de un umbral de 0.85 podemos utilizar cualquiera de los métodos propuestos, mientras que para valores iguales o menores a 0.85 podemos elegir entre WU-PALMER, JIANG-CONRATH y COSENO en cualquiera de sus variantes. De forma excepcional a esta lista podemos añadir LIN-2 para valores de

Tabla 6.31: p-valores ajustados para JIANG-CONRATH-1 (Friedman)

	t=0.9		t=1.0	
algoritmo	p sin ajustar	p_{Hommel}	p sin ajustar	p_{Hommel}
JACCARD-1	0.071861	0.143721	0.230139	0.460279
JACCARD-2	0.071861	0.143721	0.230139	0.460279
JACCARD-3	0.071861	0.143721	0.230139	0.460279
SORENSEN-1	0.071861	0.143721	0.230139	0.460279
SORENSEN-2	0.071861	0.143721	0.230139	0.460279
SORENSEN-3	0.071861	0.143721	0.230139	0.460279
MOUNTFORD-1	0.071861	0.143721	0.230139	0.460279
MOUNTFORD-2	0.071861	0.143721	0.230139	0.460279
MOUNTFORD-3	0.071861	0.143721	0.230139	0.460279
OVERLAP-1	0.071861	0.143721	0.230139	0.460279
OVERLAP-2	0.071861	0.143721	0.230139	0.460279
OVERLAP-3	0.071861	0.143721	0.230139	0.460279
RADA-1	0.071861	0.143721	0.230139	0.460279
WU-PALMER-1	0.071861	0.143721	0.230139	0.460279
WU-PALMER-2	0.071861	0.143721	0.230139	0.460279
RESNIK-1	0.071861	0.143721	0.230139	0.460279
RESNIK-2	0.071861	0.143721	0.230139	0.460279
LIN-1	0.071861	0.143721	0.230139	0.460279
LIN-2	0.071861	0.143721	0.230139	0.460279
COSENO-1	0.071861	0.143721	0.230139	0.460279
COSENO-2	0.071861	0.143721	0.230139	0.460279
COSENO-3	0.071861	0.143721	0.230139	0.460279
JIANG-CONRATH-2	1	1	1	1

Tabla 6.32: p-valores ajustados para WU-PALMER-2 (Friedman)

	t=0.6		t=0.65		t=0.7	
algoritmo	p sin ajustar	p_{Hommel}	p sin ajustar	p_{Hommel}	p sin ajustar	p_{Hommel}
JACCARD-1	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
JACCARD-2	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
JACCARD-3	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
SORENSEN-1	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
SORENSEN-2	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
SORENSEN-3	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
MOUNTFORD-1	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
MOUNTFORD-2	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
MOUNTFORD-3	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
OVERLAP-1	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
OVERLAP-2	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
OVERLAP-3	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
RADA-1	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
RESNIK-1	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
RESNIK-2	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
LIN-1	0.003445	0.027556	0.003732	0.026121	0.004728	0.033096
LIN-2	0.012419	0.086935	0.003732	0.026121	0.004728	0.033096
JIANG-CONRATH-1	0.36812	0.86108	0.36812	0.841481	0.2113	0.880765
JIANG-CONRATH-2	0.423711	0.86108	0.423711	0.841481	0.565291	0.880765
WU-PALMER-1	0.726339	0.86108	0.841481	0.841481	0.880765	0.880765
COSENO-1	0.317311	0.86108	0.32956	0.841481	0.169131	0.845657
COSENO-2	0.783316	0.86108	0.802587	0.841481	0.821979	0.880765
COSENO-3	0.86108	0.86108	0.70766	0.841481	0.548506	0.880765

Tabla 6.33: p-valores ajustados para WU-PALMER-2 (Friedman) (cont.)

	t=0.75		t=0.8		t=0.85	
algoritmo	p sin ajustar	p_{Hommel}	p sin ajustar	p_{Hommel}	p sin ajustar	p_{Hommel}
JACCARD-1	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
JACCARD-2	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
JACCARD-3	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
SORENSEN-1	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
SORENSEN-2	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
SORENSEN-3	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
MOUNTFORD-1	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
MOUNTFORD-2	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
MOUNTFORD-3	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
OVERLAP-1	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
OVERLAP-2	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
OVERLAP-3	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
RADA-1	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
RESNIK-1	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
RESNIK-2	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
LIN-1	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
LIN-2	0.003445	0.024112	0.00552	0.038641	0.006934	0.048538
JIANG-CONRATH-1	0.177016	0.708064	0.282375	0.564749	0.381574	0.670837
JIANG-CONRATH-2	0.468452	0.880765	0.282375	0.564749	0.381574	0.670837
WU-PALMER-1	0.880765	0.880765	0.617075	0.617075	0.670837	0.670837
COSENO-1	0.2113	0.709295	0.02963	0.177781	0.02963	0.11852
COSENO-2	0.531971	0.880765	0.293718	0.587436	0.02963	0.11852
COSENO-3	0.43834	0.876679	0.121142	0.391624	0.02963	0.11852

umbral de 0.6. La selección del método a utilizar, por tanto, dependerá de otros factores. Analicemos algunos de ellos.

- **Relevancia semántica:** Como se ha podido comprobar en la Tabla 6.29 la medida que tiene una mayor correlación con juicios humanos es la del Coseno, seguida de Wu-Palmer y finalmente Jiang-Conrath. Si deseamos dar mayor relevancia a la semántica a la hora de generar la ontología nos deberíamos de decantar por el método COSENO.

- **Número de clases en la ontología final:** Dependiendo del número de clases que queramos obtener utilizaremos aquellos métodos que aceptan más categorías. Las categorías adicionales, aunque no tengan efecto sobre la cobertura, pueden enriquecer la semántica de la ontología final. En la Figura 6.3 se muestra una gráfica con la media del número de nodos obtenidos según el valor de umbral para cada método. Los métodos del COSENO son los que mayor número de clases en la ontología generan, seguidos de WU-PALMER y JIANG-CONRATH. De igual forma, el ámbito de comparación afecta al número de categorías seleccionadas, de modo que la comparación con la categoría inicial (1), selecciona un número menor de categorías, seguido de la comparación con la categoría actual (2), y por la comparación con la categoría acumulada (3), que es la que más categorías selecciona. Si queremos una ontología compacta debemos usar JIANG-CONRATH-1 y si deseamos obtener un número elevado de nodos, podemos usar COSENO-2.

- **Tiempo de ejecución:** En ocasiones el tiempo de ejecución de los algoritmos puede

ser crucial, especialmente si queremos generar las ontologías en tiempo real. En la Figura 6.4 vemos una gráfica con la media de los tiempos de ejecución para todos los métodos, a lo largo de los diferentes conjuntos de datos, y para distintos valores de umbral. Como se puede observar en la gráfica los métodos del COSENO son los que mayor tiempo de ejecución necesitan, la mayor parte de este tiempo se emplea en el procesamiento del contenido de las páginas. A mayor número de categorías aceptadas, mayor número de comparaciones y por tanto de tiempo de ejecución. En la Figura 6.5 se muestra una gráfica con el comportamiento de los métodos que podemos considerar que mejoran al resto, excluyendo el COSENO que ya hemos visto que claramente tiene tiempos de ejecución mayores. Podemos observar como el tiempo de ejecución se dispara para umbrales menores de 0.625. Para el resto los métodos vemos como LIN-2 y JIANG-CONRATH-2 aparecen como los más estables. Dado que LIN-2 sólo presenta diferencias significativas a partir de valores de umbral de 0.6, en este caso emplearíamos JIANG-CONRATH-2 por ser el más rápido, y por tanto nos permitiría generar las ontologías en tiempo real.

- **Complejidad del cálculo:** Como hemos comentado a lo largo de la presente memoria, algunos de los métodos presentan más dificultad que otros a la hora de su implementación, ya que están pensados para ser utilizados en jerarquías, y deben adaptarse para el uso en grafos. Esta adaptación provoca pérdida de información en la mayoría de los casos. Si tenemos en cuenta la complejidad del proceso, debemos recordar que las medidas de Wu-Palmer y Jiang-Conrath hacen uso del *lcs* en sus cálculos. Esto implica la necesidad de transformar el grafo de categorías en una jerarquía con los inconvenientes que ello conlleva. Además, la medida de Jiang-Conrath está basada en el cálculo del contenido de la información, y por tanto ha de calcular el árbol de hipónimos. Esto se traduce en una fuente adicional de errores, ya que este árbol contiene varios nodos que por su número de hipónimos pueden considerarse raíces, lo cual desvirtúa las medidas. La creación de estas estructuras adicionales sólo se realiza una vez, y se almacena en disco. A partir de su almacenaje sólo deberá leerse en memoria en cada ocasión, aunque este proceso tiene una duración que ronda un minuto de tiempo aproximadamente, el principal inconveniente es la información perdida en la transformación del grafo. En este caso, el método del COSENO parece el más indicado ya que no necesita adaptar el grafo, basta con recorrerlo usando una política en la que no se expandan nodos ya visitados. Aunque el cálculo de los vectores de términos entraña cierta dificultad, en ningún caso es comparable a la de transformar el grafo en un jerarquía acíclica.

- **Requisitos de memoria:** Los métodos seleccionados tienen necesidad de grandes cantidades de memoria, WU-PALMER y JIANG-CONRATH deben leer el grafo de categorías. JIANG-CONRATH además debe mantener en memoria el árbol de hipónimos. Por otra parte, todas las operaciones realizadas por el método del COSENO se hacen en memoria, y se almacena la categoría de comparación. En relación con los otros métodos los requisitos de memoria son muy bajos, excepto cuando se usa la comparación con la categoría acumulada. La categoría acumulada debe mantenerse

en memoria para realizar las comparaciones y el vector de términos que la define es de gran tamaño. En general el método del COSENO consume menos memoria, excepto para COSENO-3, en este caso WU-PALMER y JIANG-CONRATH necesitan menos memoria (siempre y cuando las estructuras adicionales ya hayan sido generadas).

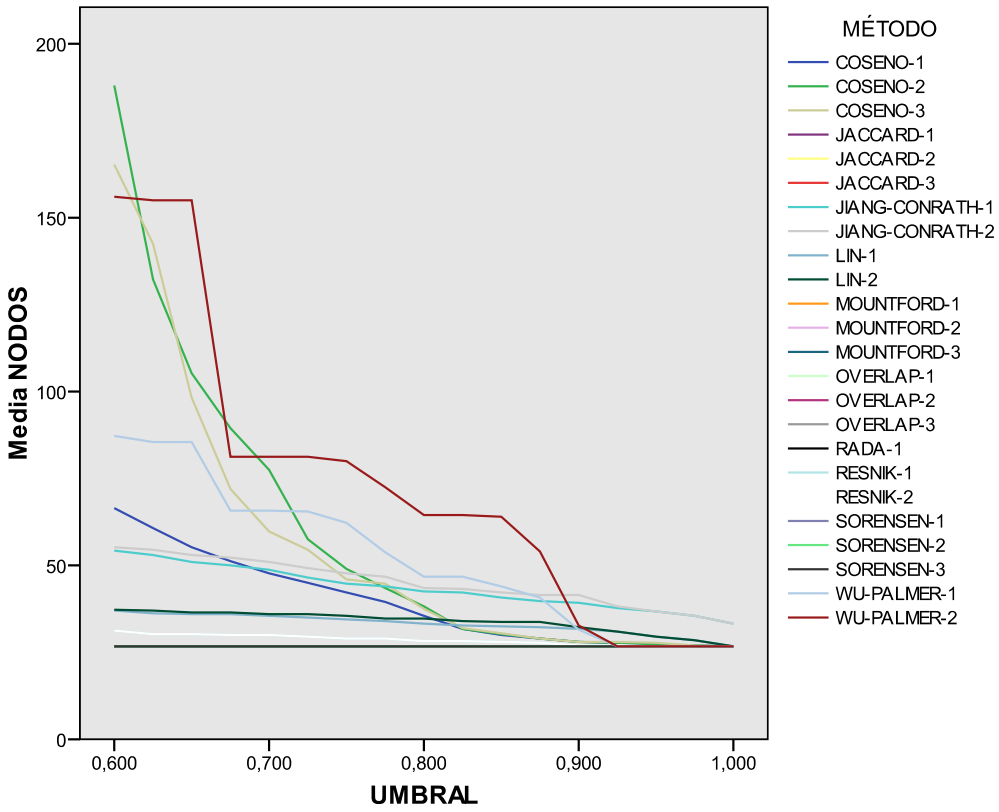


Figura 6.3: Media del número de nodos en la ontología por cada método

En general viendo los distintos escenarios que se pueden dar, creemos que los métodos del COSENO son los más adaptables, exceptuando aquellos casos en los que el tiempo de ejecución y el uso de memoria son factores críticos.

6.6. Conclusiones

A lo largo de esta sección hemos evaluado las ontologías obtenidas por medio de la extracción de semántica desde texto y la posterior extensión con Wikipedia. Hemos creado cuatro conjuntos de datos con diferentes características y hemos generado ontologías basadas en esos conjuntos para diferentes valores de umbral. Tras realizar una breve discusión de las posibilidades existentes para la evaluación de ontologías, nos hemos decantado

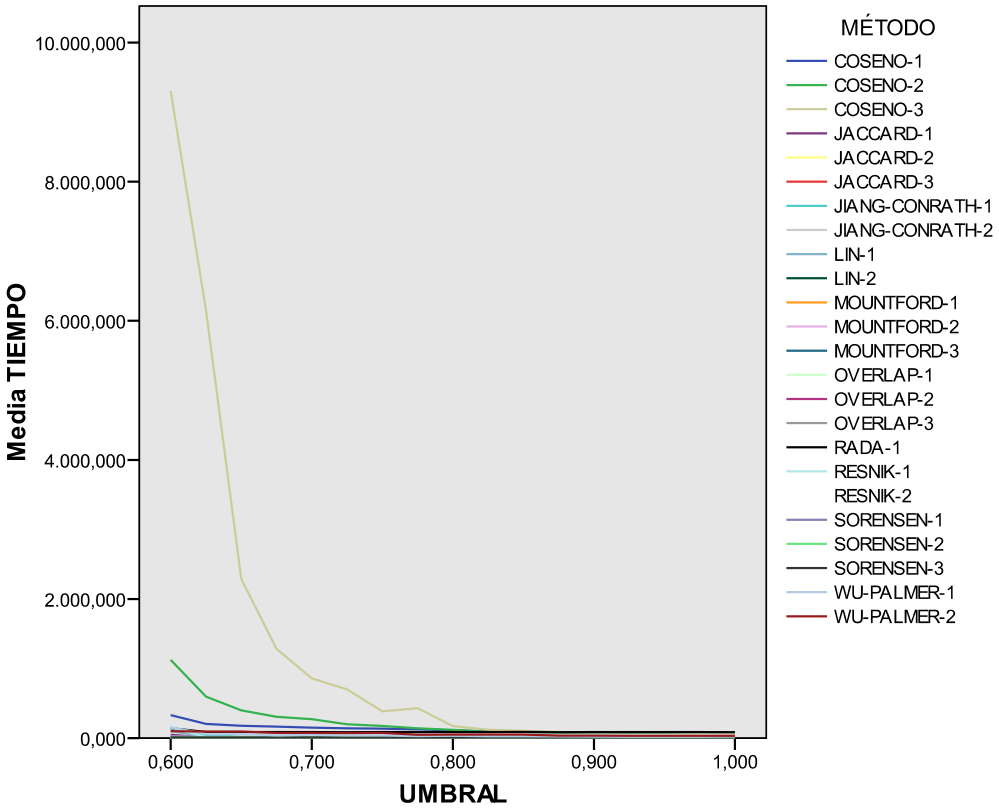


Figura 6.4: Media de los valores de tiempo en segundos

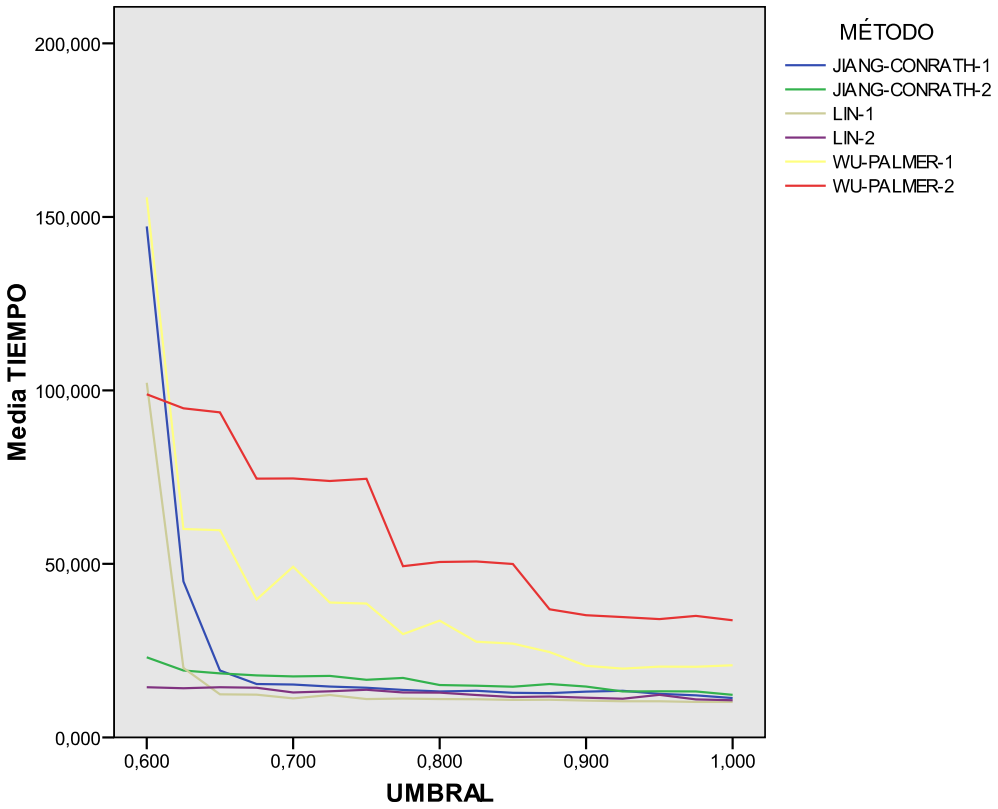


Figura 6.5: Media de los valores de tiempo en segundos

por realizar una evaluación de la relevancia semántica de las distintas medidas, una evaluación estructural, y una evaluación basada en el uso de la ontología en el contexto de una aplicación. La aplicación que hemos seleccionado, consiste en la realización de consultas basadas en conceptos a partir de las ontologías generadas, las cuales han sido anotadas con WordNet con términos de consulta adicionales para cada concepto. Las consultas se lanzarán sobre el conjunto de datos original, de tal forma que los valores de cobertura nos darán una idea de lo bien que representa la ontología los textos originales.

Tras descartar métodos que no se adaptaban bien a este tipo de evaluación tales como los basados en la exploración del grafo y Leacock-Chodorow que no está normalizado, hemos obtenido ontologías para cada uno de los 10 métodos propuestos y sus variantes, por cada uno de los 4 conjunto de datos y para umbrales en el rango $[0.6, 1]$ en incrementos de 0.25. En total hemos generado un total de 1633 ontologías para la evaluación.

Los resultados obtenidos muestran que los métodos basados en índices estadísticos no presentan resultados relevantes. Lo cierto es que en rara ocasión mejoran el caso base. Los métodos basados en similitud semántica, con la excepción de WU-PALMER, tampoco demuestran un buen comportamiento. En cuanto a los métodos basados en contenido de la información, si bien son algo mejores que los anteriores únicamente JIANG-CONRATH demuestra destacar, también lo hace LIN para valores de umbral cercanos a 0.6. Si bien los métodos del COSENO tienen un buen comportamiento su tiempo de ejecución lastra excesivamente la comparación con otros métodos, y los descarta como posibles candidatos a usar en un generación en tiempo real de las ontologías.

En general hemos mostrado que WU-PALMER, JIANG-CONRATH y COSENO, son mejores significativamente con respecto al resto de métodos para un umbral en el rango $[0.6, 0.8]$. Para valores superiores, no se puede determinar estadísticamente que existan diferencias significativas en el comportamiento de los métodos.

Así pues, se recomienda el uso de WU-PALMER cuando se desee tener un número elevado de clases en la ontología final o un tiempo de ejecución bajo. Sin embargo se debe de tener en cuenta que este método necesita de mucho procesamiento, ya que se debe convertir el grafo de categorías en un árbol, con la pérdida de información que conlleva, y un incremento en el cómputo y tiempo necesarios. Si bien este cálculo se realiza una única vez, el resto de ejecuciones se debe de leer el árbol de disco y cargarlo en memoria, lo cual también consume tiempo. Las diferentes versiones de WU-PALMER según el ámbito de comparación se diferencian esencialmente en el número de nodos y el tiempo de ejecución, siendo ambos más elevados para WU-PALMER-2.

El método de JIANG-CONRATH está indicado para escenarios en los que el número de clases a obtener en la ontología final no sea muy elevado, y el tiempo de cómputo deba ser bajo. También utiliza la conversión del grafo de categorías en árbol, y además, debe generar el árbol de hipónimos necesario para calcular el contenido de la información. Ambas estructuras deben permanecer en memoria durante la ejecución del método.

Por último el método del COSENO está indicado para situaciones en las que queramos obtener criterios de selección similares a los que realizaría un humano, puesto que este método es el que mejor correlación tiene con juicios humanos. El número de clases en las ontologías generadas es elevado. Tiene el principal inconveniente del tiempo de ejecución y la cantidad de memoria usada (especialmente para COSENO-3, en los dos otros casos no

supone un factor determinante). Sin embargo, tiene la ventaja de no necesitar estructuras adicionales como los casos anteriores.

Cabe destacar que LIN-2 aparece como otro de los algoritmos a tener en cuenta para un umbral de 0.6. Dado que éste es un valor límite y que en el resto de casos WU-PALMER, JIANG-CONRATH y COSENO muestras diferencias significativas respecto a LIN-2, podemos descartar este método y quedarnos con los otros tres, que han demostrado un comportamiento más consistente.

Queda pendiente una evaluación desde el punto de vista ontológico, pero dado el gran volumen de ontologías con el que trabajamos, esto sólo será posible si existe alguna metodología de aplicación automática.

Capítulo 7

Conclusiones y Trabajos Futuros

En este capítulo presentaremos las conclusiones derivadas del trabajo de investigación que se ha realizado, y los trabajos futuros a realizar para la obtención de resultados que complementen lo realizado hasta el momento.

7.1. Conclusiones

En este trabajo hemos tratado con el tratamiento de la semántica de la información en entornos de bases de datos relacionales, aunque por sus capacidades expresivas y de extensión nos hemos centrado en el uso de un modelo objeto-relacional.

Hemos abordado el estudio de la semántica en bases de datos a dos niveles:

- **Proporcionando semántica a las estructuras de la base de datos**.

  Para la gestión de la semántica estructural de la base de datos se ha optado por realizar una representación del conocimiento basada en ontologías.

  - De este modo hemos propuesto el uso de ontologías de dominio definidas en OWL para la descripción del nivel conceptual de esquemas de bases de datos. Esta propuesta combina diversos métodos existentes para una correcta representación de la ontología y las instancias en una base de datos, sin pérdida de la semántica. La representación de la ontología en la base de datos junto con sus instancias permite realizar consultas semánticas sobre un amplio conjunto de datos.
  - El uso de una ontología de dominio como herramienta de diseño del esquema de base de datos, facilita el trabajo de desarrollo de esquemas de almacenamiento para sistemas que se han de adaptar a la Web Semántica, al mismo tiempo que almacena la descripción de la semántica de la estructura dentro de la propia base de datos. Esto permite conocer el modelo a partir del cual se ha generado el esquema, permite procesarlo y además puede ser consultado como parte de la documentación del esquema.

- **Proporcionando semántica al contenido de la base de datos**.

  La gestión de la semántica del contenido se ha abordado desde dos vertientes, el tratamiento del contenido numérico y el tratamiento de los campos textuales.

  - *Atributos Numéricos*. Los campos numéricos de una base de datos tienen una semántica bien definida y por tanto es complicado obtener semántica adicional mas allá de la intrínseca que poseen. Mediante el uso de la lógica difusa integrada en un entorno de base de datos, hemos podido dotar de una mayor semántica a los datos numéricos representando grados de incertidumbre. La representación numérica empleando valores aproximados, distribuciones de posibilidad y rangos, permite realizar consultas sobre los datos más significativas y cercanas al usuario.
  - *Atributos Textuales*. El tratamiento de campos textuales es muy importante, en ocasiones, los textos de una base de datos no suelen ser objeto de consulta, dado que una búsqueda sintáctica no aporta relevancia alguna en la mayoría de ocasiones. La obtención de una representación semántica de los textos va a permitir realizar consultas mas complejas sobre estos, que en lugar de incluir únicamente términos, incluyan conceptos. Para el tratamiento del contenido en campos textuales, hemos realizado una propuesta para la extracción de contenido semántico a partir de textos no estructurados. Esta propuesta se ha realizado de forma genérica de tal forma que pueda ser implementada mediante el uso de diversas herramientas. En nuestro caso hemos estudiado diferentes posibilidades de implementación sugiriendo distintas herramientas en cada una de las etapas de la metodología.

    De entre las distintas etapas definidas en la metodología, hemos centrado nuestra atención en dos, representación y extensión semántica. Para la representación hemos utilizado técnicas de minería de datos y hemos empleando como representación intermedia los AP-Sets. A partir de estos se han generado ontologías básicas a modo de taxonomías, que nos han permitido obtener más información acerca de los textos, en términos de conjuntos de términos frecuentes en los textos. Mediante la representación en AP-Sets obtenemos la información relevante para el dominio, basándonos en criterios de frecuencia de aparición de los términos, y una vez obtenidos los datos relevantes obtenemos una ontología a partir de ellos.

    A partir de la representación intermedia basada en AP-Sets hemos generado ontologías con el apoyo de fuentes de conocimiento externas.

    - *Extensión semántica utilizando WordNet*. Se ha realizado una implantación de la metodología empleando AP-Sets y WordNet. Hemos utilizado WordNet en dos etapas de la metodología, preprocesamiento semántico y extensión semántica. El preprocesamiento semántico empleando WordNet, permite agrupar términos sinónimos, y determinar un representante del conjunto. Este representante se utiliza para evitar la aparición de múltiples términos que designan al mismo concepto. WordNet es una herramienta

muy apropiada para este tipo de tarea. Sin embargo, hemos podido comprobar que utilizar WordNet para extensión semántica no proporciona buenos resultados, debido a su orientación eminentemente lingüística y a la baja cobertura encontrada para términos técnicos. Esto nos ha llevado a buscar otras fuentes externas que pudiesen ofrecer mejores resultados.

- *Extensión semántica utilizando Wikipedia.* La cobertura de Wikipedia, su continua actualización y su estructura conceptual de tipo mas general nos ofrecían razones suficientes para realizar un estudio en más profundidad. Por tanto, hemos estudiado y evaluado distintos tipos de medidas de similitud y relación semántica sobre el grafo de categorías de Wikipedia.
  Para el análisis de los resultados hemos recurrido a trabajos sobre evaluación de ontologías y hemos decidido realizar una evaluación en base al uso de la ontología en una aplicación. Nos hemos decidido por esta opción ante de la imposibilidad de poder realizar una evaluación mediante expertos, dada la gran cantidad de ontologías generadas y la complejidad de alguna de estas. Por otra parte, la evaluación con respecto a una ontología de referencia no es viable debido a que la naturaleza de los textos determina las ontologías generadas, y es complicado encontrar una ontología que modele ese dominio en concreto.
  La aplicación para la evaluación de ontologías que hemos propuesto se ha basado en la consulta por conceptos y en la cobertura de dichas consultas del conjunto de datos a partir del cual se han generado las ontologías. De los estudios realizados hemos podido comprobar cómo el contenido de las categorías, o su posición relativa dentro del grafo son elementos clave para determinar la relación existente entre pares de categorías.

Finalmente, hemos presentado como material complementario una breve introducción a la bases de datos difusas y hemos realizado un repaso del estado del arte en bases de datos objeto-relacionales, se ha disertado acerca de la representación de ontologías, y se ha introducido brevemente la base de datos léxica WordNet. También se ha ofrecido una breve descripción de las herramientas creadas para llevar a cabo la implementación de las diferentes etapas descritas en la metodología general.

7.2. Trabajos Futuros

Cada una de las líneas de trabajo presentadas, suponen un acercamiento inicial y una serie de propuestas para la resolución de una serie concreta de problemas. A continuación veremos los trabajos futuros a desarrollar para cada una de ellas.

- **Representación de ontologías en bases de datos**.

  Entre nuestras futuras direcciones de trabajo, se encuentra la posibilidad de incluir un razonamiento elemental. Adicionalmente contemplamos la posibilidad de combinar distintas bases de datos creadas con el procedimiento propuesto, realizando un

proceso de mapeo y fusión de ontologías para generar la base de datos resultante. Otro aspecto interesante es la definición, evaluación y análisis de rendimiento de los procesos de concordancia semántica, adaptando las estructuras para mejorar la eficiencia en caso de ser necesario.

- **Extensión semántica de textos no estructurados**.

  En esta línea, los trabajos futuros estarán orientados a mejorar la representación ontológica de los AP-Sets, obteniendo nuevas relaciones a partir de WordNet o de otras fuentes con contenido semántico como DBPedia o ConceptNet. Otro de los trabajos que deben desarrollarse, es la aplicación y adaptación de estas técnicas a textos de una mayor longitud. También se plantea la implementación de una herramienta de consulta integrada con Protégé, que permita formular consultas semánticas que serán respondidas por la base de datos. Para poder obtener dichas respuestas se debe implementar el conjunto de procedimientos y funciones de concordancia semántica que tengan en cuenta las estructuras de almacenamiento que hemos incluido en el catálogo.

La implementación de un esquema de almacenamiento de las ontologías de consulta generadas para los campos textuales, completará el ciclo de nuestra propuesta. De este modo podremos gestionar toda la información de carácter ontológico de forma interna en la base de datos.

Apéndices

Apéndice A

Introducción

En este apéndice realizaremos una breve introducción a algunos de los principales conceptos utilizados a lo largo del presente trabajo, pero que han sido tratados de forma tangencial. Esperamos que esta información sirva de complemento y ayuda, a aquellos que se acercan por primera vez a estas materias.

Como se ha comentado anteriormente, nuestra propuesta se soporta sobre un sistema gestor de información, compuesto por una Base de Datos Objeto-Relacional Difusa. Es por esto, que veremos un primer acercamiento a la teoría de conjuntos difusos y a los conceptos básicos sobre estos. Continuaremos con una breve introducción a las bases de datos difusas en sus vertientes de implementación relacional y objeto-relacional. Finalmente realizaremos un repaso a los distintos modelos de bases de datos difusas existentes.

Debido a que nuestra propuesta incluye un tratamiento semántico de la información a través de ontologías, en consonancia con los trabajos desarrollados sobre Web Semántica, realizaremos un breve resumen sobre de ontologías, y su representación utilizando OWL.

Finalmente, este apéndice se cierra con una breve descripción de la base de datos léxica WordNet, un recurso muy utilizado para procesamiento del lenguaje natural.

A.1. Teoría de Conjuntos Difusos

A lo largo de las siguientes secciones de este apéndice, aparecerán una serie de conceptos estrechamente relacionados con la teoría de conjuntos difusos. Es por esto que a continuación, proporcionaremos una visión general sobre las nociones elementales de esta teoría.

A.1.1. Conjuntos Difusos

En la teoría de conjuntos clásica, una función característica asigna un valor de 0 o 1, a cada individuo del conjunto universal dependiendo de si se encuentra, o no, dentro del conjunto. Esta función, se puede generalizar de modo que los valores asignados a los elementos se encuentren dentro de un rango especificado, que indique el grado de pertenencia de esos elementos en cuestión. Valores más altos determinan grados de pertenencia mayores al conjunto. Esta función se llama *función de pertenencia* y el conjunto definido, *conjunto difuso*.

Sea X el conjunto universal, x los elementos genéricos de ese universo, y A un subconjunto difuso de X que no tiene límites definidos.

El conjunto de valoración de A es un intervalo, por lo que A se llama *conjunto difuso* (Zadeh, 1965).

La función de pertenencia $X \rightarrow [0,1]$, determina el grado de pertenencia de cada elemento x a A, asignándole un valor en el intervalo [0,1] donde 0 indica la no pertenencia al conjunto y 1 la total pertenencia al mismo.

Las funciones de pertenencia suelen tener formas muy variadas, pero habitualmente las representaciones más utilizadas son la triangular y la trapezoidal.

A se caracteriza completamente por un par.

$$A = \{(x, \mu_A(x)), x \in X\}.$$

Una notación más conveniente fue propuesta por Zadeh. Cuando X es un conjunto finito $\{x_1, ..., x_n\}$, un conjunto difuso en X se expresa como:

$$A = \mu_A(x_1)/x_1 + \ldots + \mu_A(x_n)/x_n = \sum_{i=1}^{n} \mu_A(x_i)/x_i.$$

Cuando X no es finito escribimos:

$$A = \int_x \mu_A(x)/x.$$

Dos conjuntos difusos son iguales ($A = B$) si:

$$\forall x \in X,\ \mu_A(x) = \mu_B(x).$$

A.1.2. Conceptos básicos sobre conjuntos difusos

Una vez definido el concepto de conjunto difuso, vamos a ver otros conceptos básicos de la teoría de conjuntos difusos. La mayoría de los conceptos presentados son extensiones y generalizaciones de conceptos de conjuntos clásicos, pero otros son únicamente aplicables al marco difuso.

El *soporte* de un conjunto difuso A es un subconjunto regular de X, que contiene todos los elementos cuyo grado de pertenencia a A, es mayor que 0.

$$\text{supp } A = \{x \in X,\ \mu_A(x) > 0\}.$$

Los *puntos de cruce* de A, son aquellos que pertenecen al conjunto con el mismo grado con el que no pertenecen.

$$\text{crossover points } A = \mu_A(x) = 1/2.$$

La *altura* de A es el mayor grado de pertenencia de los elementos del conjunto.

$$hgt(A) = \sup_{x \in X} \mu_A(x).$$

A se dice *normalizado* si su altura es 1.

$$\exists x \in X,\ \mu_A(x) = 1 \Rightarrow hgt(A) = 1.$$

El *núcleo* de A son todos aquellos elementos cuyo grado de pertenencia al conjunto es 1. Si el núcleo de A es no vacío, A está normalizado y la altura del conjunto es 1.

$$x \in X,\ \mu_A(x) = 1.$$

El *conjunto vacío* se define de la siguiente manera:

$$\forall x \in X,\ \mu_\emptyset(x) = 0.$$

Del mismo modo, podemos suponer que el grado de pertenencia de los elementos x con respecto al conjunto universal es 1.

$$\forall x,\ \mu_X(x) = 1.$$

Un α-corte de un conjunto difuso A, es un conjunto clásico A_α que contiene todos los elementos del conjunto universal X que tienen un grado de pertenencia en A mayor o igual que el valor especificado de α.

$$A_\alpha = \{x \in X \,|\mu_A(x) \geq \alpha\}.$$

El *α-corte estricto* se define como...

$$A_\alpha = \{x \in X \,|\mu_A(x) > \alpha\}.$$

El conjunto de todos los niveles $\alpha \in [0,1]$ que representan distintos α-cortes de un conjunto difuso dado A, se conoce como *conjunto de nivel* de A.

$$\Lambda_A = \{(\alpha \,|\mu_A(x) = \alpha), x \in X\}\,.$$

Cuando X es un conjunto finito, la *cardinalidad escalar* $|A|$ de un conjunto difuso A en X se define como la sumatoria de los grados de pertenencia de todos los elementos en ese conjunto A.

$$|A| = \sum_{x \in X} \mu_A(x).$$

$||A|| = |A|/|X|$ es la *cardinalidad relativa*, puede interpretarse como la proporción de elementos de X que están en A.

También existe la *cardinalidad difusa*, definida como un número difuso en lugar de hacerlo con un número real, como en el caso de la cardinalidad escalar. Cuando A tiene soporte finito, su cardinalidad difusa es...

$$|A|_f = \sum \alpha/|A_\alpha| = \{(n,\alpha), n \in \mathbb{N} \; and \; \alpha = \sup\{\lambda, |A_\lambda| = n\}\}\,.$$

La noción de *convexidad* puede generalizarse a conjuntos difusos de un universo X, que asumiremos como espacios reales Euclídeos N-dimensionales. Un conjunto difuso A es convexo, si solo si, sus α-cortes son convexos. Una definición equivalente es...

$$\forall x_1 \in X, \forall x_2 \in X, \forall \lambda \in [0,1],$$
$$\mu_A(\lambda x_1 + (1-\lambda)x_2) \geq \text{mín}(\mu_A(x_1), \mu_A(x_2)).$$

Aunque el rango de valores [0,1] es el más utilizado habitualmente para representar grados de pertenencia, en realidad puede usarse cualquier conjunto arbitrario con un orden parcial o total. Los elementos de este conjunto no necesariamente han de ser números, basta con que el orden entre ellos pueda interpretarse como representación de valores diferentes de grado de pertenencia. Esta función de pertenencia generalizada tiene la forma

$$\mu_A : X \to L,$$

donde L representa cualquier conjunto que al menos está parcialmente ordenado. Los conjuntos difusos definidos por esta función de pertenencia se llaman *conjuntos L-difusos*. La denominación de L viene del vocablo inglés *lattice* (retículo). Este tipo de conjuntos serán los que nos permitirán representar valores lingüísticos.

Hemos visto en este apartado los conceptos básicos relacionados con conjuntos difusos, tal como se formularon originalmente. Con posterioridad la definición de conjunto difuso se ha ampliado. Así pues, aparecen los conjuntos difusos de tipo 2 que son aquellos en los que los grados de pertenencia son difusos, y también los conjuntos probabilísticos que son una variable aleatoria sobre un espacio probabilístico.

A.2. Bases de datos Difusas

En esta sección se realiza una breve introducción a las bases de datos difusas, los modelos en que se basan y las distintas formas en que se introducen los conceptos difusos en dichos modelos.

Nos centraremos principalmente en los dos modelos más extendidos y que mayor proyección de futuro tienen *a priori*, concretamente analizaremos los modelos de bases de datos relacional difuso y objeto-relacional difuso. Para cada uno de estos modelos vamos a realizar una introducción general, sin entrar en detalles más específicos de los modelos, ya que más adelante se dedicará una sección a analizar pormenorizadamente las distintas propuestas para cada uno de los modelos.

En esta sección también se presentarán los diferentes mecanismos de extensión que ofrecen las distintas bases de datos clásicas, de los cuales se puede hacer uso con el objetivo de convertirlas en bases de datos difusas.

A.2.1. BD Difusas Relacionales

El campo de estudio de las bases de datos relacionales clásicas ha sido ampliamente explotado y se está llegando a un punto donde las investigaciones en este área se centran, ya no tanto en nuevos desarrollos que puedan aportar mejoras, si no en optimizar al máximo los modelos existentes. Prácticamente, el ámbito de estudio de las bases de datos relacionales ha quedado relegado a la mejora de la eficiencia, los sistemas distribuidos y el procesamiento de transacciones.

En la actualidad se tiende más a centrar las investigaciones en lo que es el nuevo modelo Objeto-Relacional, que propone unir las mejores características de los modelos de bases de datos relacional y orientado a objetos.

La aparición de los sistemas relacionales difusos ha conseguido reactivar el campo de estudio de las bases de datos relacionales. El principal objetivo de los diferentes modelos de bases de datos relacionales difusas (BDRD) es el almacenamiento de la información imprecisa con la finalidad de poder operar con ésta de forma coherente. Existen una gran variedad de modelos de BDRD y no todos ellos operan con la imprecisión a los mismos niveles, vamos a ver de forma breve las distintas formas de tratar la imprecisión en las bases de datos relacionales.

1. *Valores difusos en los atributos.* En los atributos se van a almacenar diferentes tipos de datos difusos con los que se podrá operar. Dependiendo del tipo del atributo admitirá un determinado tipo de valores difusos (valores aproximados, rangos, distribuciones de posibilidad, etiquetas lingüísticas, etc...). Algunos modelos aceptan un conjunto reducido de estos tipos y otros son más generales, trataremos en profundidad este aspecto cuando nos centremos en el análisis de los trabajos previos realizados en este campo.

2. *Valoración difusa de un atributo.* Cada valor de un atributo tiene asociada una valoración difusa en el intervalo [0,1], que indica en qué grado el valor es difuso. Este tipo de valores también se emplean en varios modelos.

3. *Valoración difusa de una tupla.* Este caso es bastante parecido al caso anterior, salvo que se indica el grado difuso para toda la tupla.

4. *Valoración difusa de un conjunto de atributos.* Este caso es una mezcla de los dos casos anteriores, existe un único grado que representa a un grupo de atributos de la tupla, no a todos.

Hemos hablado acerca de la valoración difusa de los distintos elementos que forman parte de una relación. Pero también es importante hacer hincapié en que esa valoración puede ser interpretada de diferentes formas.

El grado de un valor difuso, se va a representar como un valor dentro del intervalo [0,1], tal y como se ha comentado con anterioridad. Veamos las posibles interpretaciones de un grado de valoración difuso.

- *Grado de cumplimiento:* Indica que se cumple una propiedad con el grado indicado. Un cumplimiento de 0 indica que la propiedad no se cumple, y un cumplimiento de 1 indica que la propiedad se cumple totalmente. Suele emplearse para la satisfacción de condiciones difusas con un umbral mínimo.

- *Grado de pertenenecia:* Según esta interpretación, se indica el grado de pertenencia de un objeto a un conjunto. Así pues, un grado de pertenencia para una tupla, indica en que grado pertenece esa tupla a la relación.

- *Grado de incertidumbre:* Indica la certeza con la que se conoce un dato determinado. Un valor de incertidumbre 0 indica que no se conoce el dato, y un valor de incertidumbre 1 indica que se conoce con total seguridad el dato. Valores intermedios indican distintos grados de certidumbre acerca de un dato. Puede verse como un indicador de la veracidad de un dato.

- *Grado de posibilidad:* Indica en que medida es posible cada valor del dominio considerado en la descripción de un suceso almacenado.

  Indica la posibilidad de la información que está almacenada. Ante el valor de un suceso almacenado, este grado indicaría la posibilidad de aparición de ese valor una vez producido el suceso.

- *Grado de importancia:* Se utiliza para dar mayor importancia a unos atributos que a otros. Se puede utilizar a la hora de ordenar y evaluar una lista de condiciones que se quieren satisfacer sobre una relación, dando mayor peso a las más importantes.

Como hemos podido ver, la gran variedad de posibles modelos de representación de valores difusos y las diferentes interpretaciones de los grados, hacen que los posibles modelos de BDRD sean numerosos. Sin embargo, en esta ocasión, nos centraremos en aquellos que han tenido un mayor impacto, y que se han mantenido vigentes a lo largo del tiempo.

A.2.2. BD Difusas Objeto-Relacionales

Las necesidades de almacenamiento de datos cada vez son más complejas. En la actualidad, la simpleza y eficacia de los sistemas relacionales no puede proporcionar las capacidades necesarias para el almacenamiento y manipulación de información compleja. Ante la necesidad de mejorar el modelo relacional y adaptarlo a las nuevas necesidades de almacenamiento, se propuso liberar al modelo relacional de la restricción de que los dominios de los atributos fuesen atómicos, por lo que las relaciones no tenían por qué estar en primera forma normal (*non-first normal form* NFNF o NF2). Llevando este planteamiento hasta sus últimas consecuencias se puede definir una tabla con atributos cuyos valores sean tablas completas.

Esas bases de datos extensibles han evolucionado hacia lo que hoy en día conocemos como Tecnología Objeto-Relacional (OR), en la cual se combinan las nociones de extensibilidad, el modelo relacional y el orientado a objetos. El modelo objeto-relacional se define como una extensión de objetos del modelo relacional, permitiendo la definición de nuevos tipos y de relaciones entre estos.

Las bases de datos objeto-relacionales combinan las características de las bases de datos relacionales (modelo de datos, seguridad, concurrencia, lenguaje de alto nivel, etc...) con los principios de la orientación a objetos (encapsulamiento, polimorfismo, herencia, etc...). Estas bases de datos ofrecen la posibilidad de definir clases o tipos abstractos de datos, pudiendo crear tipos de datos definidos por el usuario (UDT).

Precisamente las capacidades de extensión y de definición de tipos de datos de usuario, son las que van a facilitar y permitir la extensión de una base de datos objeto-relacional hacia un modelo objeto-relacional difuso.

La idea tras las bases de datos objeto-relacionales difusas (BDORD) es la creación de una jerarquía de tipos de datos difusos definidos por el usuario con los que se pueda operar, así como la definición de métodos y operadores que permitan su manipulación. Existen varios trabajos que se centran en la implementación de una base de datos difusa sobre el modelo objeto-relacional, los veremos más adelante cuando se analicen los trabajos previos realizados en este campo.

A.2.3. Mecanismos de extensión de las BD existentes

El coste asociado a la creación de una base de datos difusa desde cero puede ser bastante elevado, por no hablar de lo complejo de la tarea. Una alternativa adecuada es la utilización de los mecanismos de extensión que ofrecen las bases de datos actuales. Extendiendo las bases de datos existentes, pueden crearse bases de datos difusas que aprovecharán las capacidades ya implementadas en la base de datos anfitriona, ahorrando una gran cantidad de esfuerzo y tiempo en el desarrollo.

Los sistemas relacionales pueden extenderse empleando los mecanismos proporcionados para ello, como son la creación de funciones y procedimientos definidos por el usuario, agregadores, operadores, etc... A partir de una base de datos relacional, podría crearse una base de datos relacional difusa, creando procedimientos y funciones de usuario que permitiesen dar tratamiento a los elementos difusos que se van a utilizar. Es bastante

probable que sea necesario modificar algunas de las tablas del catálogo del sistema, para permitir el almacenamiento de algunos elementos más complejos.

Los sistemas objeto-relacionales se extienden a través de los mecanismos que proporcionan los Tipos de Datos definidos por los Usuarios (UDTs) mediante los cuales pueden crearse tipos completamente nuevos y definir su comportamiento mediante la implementación de métodos que manipularán estos nuevos tipos de datos. Los mecanismos de herencia, encapsulamiento y polimorfismo, van a ser de gran utilidad a la hora de definir una jerarquía de tipos totalmente nueva, que va a contener los métodos necesarios para la manipulación de los nuevos tipos creados. Este tipo de bases de datos son por definición altamente extensibles, y por tanto será mas sencillo crear nuevos tipos y extender el lenguaje SQL que emplean, para dar cabida a los nuevos tipos definidos.

Empleando los mecanismos de extensión ofrecidos por las diferentes bases de datos, se puede construir sobre ellas extensiones para el tratamiento de datos difusos. La complejidad en cada caso dependerá de las capacidades de extensión reales de cada uno de los sistemas.

A.3. Modelos de Bases de Datos Difusos

A modo de complemento de la información presentada en el cuerpo principal de la memoria, en esta sección realizaremos un breve repaso a la evolución de las bases de datos difusas, desde sus primeros prototipos basados en el modelo relacional, a los más actuales basados en el modelo objeto-relacional. Como se ha comentado con anterioridad, sugerimos la integración del sistema propuesto dentro de un entorno de Bases de Datos Objeto-Relacionales Difusas dadas las posibilidades de interpretabilidad semántica que estas ofrecen, sobretodo en la formulación de consultas. Comenzaremos con un breve repaso de los trabajos que más impacto han causado en el ámbito de las bases de datos difusas relacionales y posteriormente continuaremos con aquellos que han hecho lo mismo en el ámbito de bases de datos objeto-relacionales difusas.

A.3.1. Modelos de Bases de Datos Relacionales Difusas

Como vimos en la Sección A.2, las bases de datos relacionales difusas son un modelo de base de datos muy útil en los sistemas de información, y se han encontrado una gran variedad de aplicaciones para ellas.

Los datos difusos se han usado para modelar información imprecisa en bases de datos desde que Zadeh introdujo el concepto de conjuntos difusos [Zadeh, 1965], como consecuencia, el modelo tradicional de base de datos relacional se ha extendido para poder manejar este nuevo tipo de datos. La mayoría del trabajo en este área se ha centrado en la extensión del modelo básico y el lenguaje de consulta, para permitir la representación y recuperación de datos imprecisos. Un gran número de temas relacionados, tales como dependencias de datos, consideraciones de implementación, etc... han sido investigados en varios modelos de bases de datos relacionales difusas.

A continuación, vamos a realizar un breve repaso a los modelos mas importantes presentes en la literatura.

Modelo de Buckles-Petry

Este modelo propuesto en [Buckles and Petry, 1982b, Buckles and Petry, 1982a], fue el primer modelo en emplear relaciones de similitud en el modelo relacional.

Los tipos de dominios soportados por el modelo son:

- *Conjunto finito de escalares.*

  Clase={Piso, Apartamento, Ático}.

  Un inmueble puede ser de clase *Piso*, *Apartamento* o *Ático*.

- *Conjunto finito de números.*

  Antigüedad={33, 34, 35}.

  La Antigüedad del inmueble son 33, 34 o 35 años.

- *Conjunto de números difusos.*

  Precio={Muy barato, Barato}.

  El precio del inmueble es *Muy barato* o *Barato*.

El significado de los valores es disyuntivo, es decir, se indican los posibles valores o *interpretaciones* que existen para un determinado atributo.

Por cada dominio existente, se define una *Relación de Similitud* para poder representar y manejar la imprecisión. Esta relación, establece en que grado o medida se parecen entre sí los diferentes elementos del dominio sobre el que se define. Los valores de similitud están comprendidos en el intervalo [0,1], indicando el valor 1 la igualdad de los elementos, y el valor 0 una diferencia absoluta. La relación de similitud de los elementos de un dominio dado es definida por el usuario.

Cuando se realiza una consulta, el proceso es el de solicitar las tuplas que cumplen unas determinadas condiciones para unos umbrales de similitud dados. Las tuplas de la relación se agrupan en clases de equivalencia, según las relaciones y umbrales de similitud definidos. Finalmente, sobre esas clases de equivalencia, operarían los operadores clásicos del álgebra relacional.

Este modelo presenta algunas carencias, ya que no consigue modelar muchos de los aspectos difusos de la información. La definición de las tuplas implica albergar varios valores dentro de un atributo, lo cual impide la atomicidad en la representación de la información. También es posible extraer varias interpretaciones del resultado de una consulta. Y además no se puede garantizar la integridad de la base de datos.

Pese a las deficiencias iniciales indicadas, no podemos olvidar que este trabajo es un referente esencial en cuanto a la definición de bases difusas relacionales y que ha impulsado toda la investigación posterior abriendo un nuevo frente en el estudio de las bases de datos relacionales.

Modelo de Prade-Testemale

Este modelo propuesto por Prade y Testemale [Prade and Testemale, 1984], permite incorporar lo que se denominan datos *incompletos* o *inciertos* en el ámbito de la Teoría de la Posibilidad.

Supongamos A un atributo cuyo dominio es D. Todo el conocimiento disponible acerca del valor que toma A para un objeto x puede ser representado mediante una distribución de posibilidad $\pi_{A(x)}$ sobre $D \cup \{e\}$, donde e es un elemento extra que denota el caso en que A no se aplica a x. En otras palabras, $\pi_{A(x)}$ es una función que va de $D \cup \{e\}$ al intervalo [0,1].

Con esta notación es posible representar elementos conocidos precisos, elementos desconocidos (pero aplicables), elementos no aplicables (indefinidos), elementos totalmente desconocidos (nulos), rangos, distribuciones de posibilidad y etiquetas lingüísticas. La estructura de los datos es similar a la empleada en el modelo de Umano-Fukami.

Este modelo emplea medidas de *posibilidad* y *necesidad* para la satisfacción de las condiciones establecidas en la consulta.

Supongamos $P(U)$ el conjunto de los subconjuntos de U. De forma general, una medida de posibilidad Π puede construirse a partir de una distribución de posibilidad π, que es una función de U a [0,1], del siguiente modo:

$$\forall A \in P(U), \quad \Pi(A) = \sup_{u \in A} \pi(u)$$

Una medida de necesidad N puede construirse a partir de a partir de Π conforme a la expresión:

$$\forall A \in P(U), \quad N(A) = 1 - \Pi(\overline{A})$$

La necesidad de un evento, corresponde con la imposibilidad del evento opuesto, por lo que se establece una relación entre posibilidad y necesidad. La posibilidad y necesidad de un evento difuso se definen de la siguiente forma:

$$\forall A \in \tilde{P}(U), \quad \Pi(A) = \sup_{u \in U} min(\pi(u), \mu_A(u))$$

$$\forall A \in \tilde{P}(U), \quad N(A) = \inf_{u \in U} max(1 - \pi(u), \mu_A(u))$$

Las distribuciones de posibilidad deben estar normalizadas para que puedan ser comparadas usando comparadores de posibilidad o necesidad, pudiendo así efectuar una determinada selección sobre una relación del modelo.

Modelo de Umano-Fukami

Uno de los primeros modelos de bases de datos relacionales difusas que presentaba distribuciones de posibilidad fue el presentado por Umano, Fukami, et al [Fukami et al., 1979, Umano, 1982].

La principal diferencia con otros modelos, es en la forma en que se concibe la información con valores desconocidos y aplicables. Se define la información *no aplicable* como una distribución de posibilidad sobre el dominio considerado, en la que cada valor de dominio aparece con un valor de posibilidad 0.

Siendo D un universo de discurso de $A(x)$ y $\pi_{A(x)}(d)$ la posibilidad de que $A(x)$ tome el valor u en U, un valor desconocido pero aplicable (*unknown*) se representa como:

$$Unknown : \pi_{A(x)}(d) = 1 \ \forall d \in D$$

Para los valores no aplicables (*undefined*) se define un caso especial de distribución de posibilidad, representada como:

$$Undefined : \pi_{A(x)}(d) = 0 \ \forall d \in D$$

Para representar la información que no se sabe si es o no aplicable, se emplea el valor especial NULL.

$$NULL = \{1/Unknown, 1/Undefined\}$$

Este modelo representa dos tipos de ambigüedad:

1. *Ambigüedad en los valores de los atributos:* Los valores de atributos que se almacenan en las relaciones pueden ser distribuciones de posibilidad.

2. *Ambigüedad en la asociación de valores:* Se asigna a cada tupla un grado de pertenencia a la relación.

El modelo define operadores básicos del álgebra relacional tales como unión, diferencia, producto cartesiano, proyección, selección. Adicionalmente define otros operadores no básicos como intersección, reunión y división.

El resultado de las consultas se divide en tres subconjuntos:

- Tuplas que claramente satisfacen la consulta.
- Tuplas que posiblemente satisfacen la consulta.
- Tuplas que claramente no satisfacen la consulta.

Este modelo es uno de los grandes referentes en el campo de la bases de datos difusas relacionales.

Modelo de Zemankova-Kandel

En [Zemankova-Leech and Kandel, 1984, Zemankova-Leech and Kandel, 1985], se define el modelo propuesto por Zemankova y Kandel. En este modelo, la manipulación de datos se basa en el álgebra relacional. El modelo está compuesto de tres partes.

1. Una base de datos de valores (VDB), donde se organizan los datos de forma similar al resto de modelos.

2. Una base de datos explicativa (EDB), donde se almacenan las definiciones para los subconjuntos difusos y relaciones difusas. Es donde se explica cómo calcular el grado de ajuste de un dato determinado con una consulta definida por el usuario.

3. Un conjunto de reglas de traducción que se emplean para el manejo de adjetivos, etc.

Los dominios definidos en el modelo pueden ser de los tipos siguientes:

1. Conjuntos escalares discretos

2. Conjuntos de números discretos, finitos o infinitos (limitados por el tamaño máximo de palabra y la precisión del computador)

3. El intervalo unidad [0,1]

El valor de los atributos puede ser:

1. Escalares simples o números.

2. Una secuencia (lista) de escalares o números.

3. Una distribución de posibilidad sobre valores de un dominio escalar o numérico.

4. Un número real del intervalo [0,1] (función de distribución de posibilidad o pertenencia).

5. Valor nulo (NULL).

Podemos observar como el tipo 3, es el más expresivo de los tipos de valores de atributos, el resto de tipos son casos especiales de éste, pero se clasifican por separado para mayor claridad y facilidad de implementación.

En el modelo se define la recuperación de información en base a consultas, según medidas de posibilidad y certeza.

Sea F un subconjunto difuso de U caracterizado por su función de pertenencia μ_F, y A un atributo que toma valores en U. La medida de posibilidad de F se define como:

$$p_A(F) = \sup_{u \in D} \{\mu_F(u) \cdot \pi(u)\}$$

Considerando que el requisito de normalización de las funciones de pertenencia de los conjuntos difusos es poco apropiado para la representación de datos que utiliza la BDRD, se introduce una nueva medida de *certeza*.

$$c_A(F) = \underset{u \in D}{max} \{0, inf\ \{\mu_F(u) \cdot \pi_A(u)\}\}$$

La interpretación de esta nueva medida de certeza es la del grado de certeza con el que A pertenece a F, o es definida por F. Uno de los inconvenientes de esta nueva medida, es que no existe una relación directa entre *certeza* y *posibilidad*, de la misma forma que había una relación entre *posibilidad* y *necesidad*.

El resultado de una consulta se representa en forma de relaciones difusas las cuales contienen dos campos en los que recogen los valores de posibilidad y certeza para cada tupla. Sobre las relaciones se pueden establecer umbrales mínimos de cumplimiento que han de satisfacer las tuplas recuperadas.

La principal ventaja de este modelo, es que muestra una gran flexibilidad en la manipulación y evaluación de información difusa, apoyándose en la libertad que ofrece la selección de un comparador difuso o de otro, y en el control del grado de satisfacción de las condiciones individuales de una consulta. Además, la cantidad de tipos de datos con los que se opera es bastante amplia en comparación con otras propuestas.

Modelo GEFRED

El modelo GEFRED (*GEneralized model for Fuzzy RElational Databases*) descrito en [Medina et al., 1994], fue propuesto por Medina, Pons y Vila en 1994. Este modelo constituye una síntesis de los modelos anteriores para tratar el problema de representación y tratamiento de la información difusa mediante bases de datos relacionales.

El modelo se basa en la definición de lo que llama *Dominio Difuso Generalizado* (D_G) y *Relación Difusa Generalizada* (R_{FG}), que incluyen a los dominios y las relaciones clásicas respectivamente.

Si U es el universo de discurso, $\tilde{P}(U)$ es el conjunto de todas las distribuciones de posibilidad definidas sobre U, incluidas las que definen los tipos *Unknown*, *Undefined* y *NULL*. Entonces se define el Dominio Difuso Generalizado como $D_G \subseteq \tilde{P}(U) \cup NULL$

Con estos dominios difusos podemos representar los siguientes tipos de datos:

1. Un escalar simple.
   $Superficie = Grande$, representado por la distribución de posibilidad $1/Grande$.
2. Un número simple.
   Antigüedad= 20, representado mediante la distribución de posibilidad 1/20.
3. Un conjunto de posibles asignaciones mutuamente excluyentes de escalares.
   $Luminosidad = \{luminoso, oscuro\}$ representado como $\{1/luminoso, 1/oscuro\}$
4. Un conjunto de posibles asignaciones mutuamente excluyentes de números.
   $Habitaciones = \{3, 4\}$ representado como $\{1/3, 1/4\}$
5. Una distribución de posibilidad en el dominio de los escalares.
   $ClaseInmueble = \{0.8/Apartamento, 0.65/Piso\}$
6. Una distribución de posibilidad en el dominio de los números.
   $Precio = \{0.1/90000, 1.0/120000, 0.7/125000\}$ números difusos o etiquetas lingüísticas.
7. Un número real perteneciente al intervalo [0,1] representando grados de cumplimiento.
   Conservación= 0.9
8. Un valor desconocido (Unknown).
   $Unknown = \{1/u : u \in U\}$
9. Un valor no definido (Undefined).
   $Undefined = \{0/u : u \in U\}$
10. Un valor nulo (NULL).
    $NULL = \{1/Unknown, 1/Undefined\}$

La *Relación Difusa Generalizada* (R_{FG}), viene dada por un par de conjuntos: *cabecera* (H) y *cuerpo* (B), $R = (H, B)$, definidos como sigue:

- *Cabecera*: consiste en un conjunto fijo de ternas *atributo-dominio-compatibilidad*, donde el atributo de compatibilidad es opcional.

  $$H = \{(A_{G1} : D_{G1}[, C_{A_{G1}}]), (A_{G2} : D_{G2}[, C_{A_{G2}}]), \ldots, (A_{Gn} : D_{Gn}[, C_{A_{Gn}}])\}$$

  Donde cada atributo A_{Gj} tiene un dominio subyacente difuso $D_{Gj} (j = 1, 2, \ldots, n)$ que no es necesariamente diferente de los otros, y $C_{A_{Gj}}$ es un *atributo de compatibilidad* que toma valores en [0,1].

- *Cuerpo*: consiste en un conjunto de tuplas difusas generalizadas distintas. Cada tupla está compuesta por un conjunto de ternas *atributo-valor-grado*, donde el grado de compatibilidad es opcional.

$$B = \{(A_{G1} : \tilde{d}_{i1}[, c_{i1}]), (A_{G2} : \tilde{d}_{i2}[, c_{i2}]), \ldots, (A_{Gn} : \tilde{d}_{in}[, c_{in}])\}$$

  Donde $(i = 1, 2, \ldots, m)$, siendo m el número de tuplas en la relación, donde $\tilde{d}_{ij}$ representa el valor de dominio que toma la tupla i sobre el atributo A_{Gj} y $c_{ij} \neq 0$ es el grado de compatibilidad asociado a ese valor.

El modelo también redefine los operadores del Álgebra Relacional en la llamada *Álgebra Relacional Difusa Generalizada*. Las operaciones clásicas de *unión*, *intersección*, *producto cartesiano*, *diferencia*, *proyección*, *selección*, *reunión* y *división*, se extienden para poder operar sobre la R_{FG} de forma coherente.

A.3.2. Modelos de Bases de Datos Objeto-Relacionales Difusas

En el campo de las bases de datos, los modelos objeto-relacionales han adquirido una gran importancia. Las capacidades de estos modelos para manipular datos complejos, los convierten en elementos clave y una extensión natural a los sistemas de bases de datos relacionales. En el momento en que los modelos relacionales estaban establecidos, comenzaron a aparecer una serie de trabajos que buscaban la extensión de estos modelos para que pudiesen manejar información difusa. Con los modelos objeto-relacionales la tarea del manejo de información difusa se simplifica, ya que estos están preparados para admitir extensiones de manejo de datos complejos.

En la actualidad existen diversos trabajos que tratan acerca de los posibles modelos para extender un sistema objeto-relacional dotandolo de capacidades de manejo de datos difusos.

Modelo de Medina-Galindo-Berzal-Serrano

Este modelo [Medina et al., 2002], emplea las capacidades de las bases de datos objeto-relacionales para crear jerarquías de tipos, tipos definidos por el usuario y operadores para modelar dominios difusos. Se utilizan como base los tipos propuestos por el modelo GEFRED [Medina et al., 1994], para bases de datos relacionales difusas.

Uno de los principales inconvenientes a la hora de formular el modelo, es la no existencia de un modelo teórico para bases de datos objeto-relacionales, del mismo modo que existe uno para bases de datos relacionales.

El *Dominio Generalizado* (FGD) es el elemento principal del modelo. En base a este dominio generalizado se establece una jerarquía de tipos. El FGD se considera como un super-tipo no instanciable que contiene todas las propiedades y operadores que existen para cualquier dato difuso. Es en este tipo donde se define el operador de igualdad difusa FEQ.

FGD tiene dos subtipos *Not Ordered Underlying Domain Data* (FUDD) que representa distribuciones de posibilidad sobre dominios subyacentes no ordenados, y *Ordered Underlying Domain Data* (FODD) que representa distribuciones de posibilidad sobre dominios

subyacentes ordenados, este tipo incluye nuevos comparadores difusos tanto de necesidad como de posibilidad.

FODD define como subtipos *Trapezoidal Data* (FTD) que representa distribuciones de posibilidad trapezoidales (valores aproximados, intervalos y etiquetas lingüísticas), *Crisp* que representa los valores escalares del SGBD anfitrión y *Other* que expresa los datos difusos empleando funciones de pertenencia de diversas formas.

FUDD define como subtipo *Scalar* que representa los datos escalares del SGBD anfitrión que no tienen definida una relación de orden entre ellos.

Finalmente se propone una implementación del modelo sobre el SGBD *Oracle 9i*.

Modelo de Cubero-Marín-Medina-Pons-Vila

Este modelo [Cubero et al., 2004] define una jerarquía de tipos similar a la del trabajo anterior, y además añade nuevos tipos tales como colecciones difusas y objetos difusos.

La figura A.1 muestra la jerarquía de tipos definidos para el modelo objeto-relacional difuso.

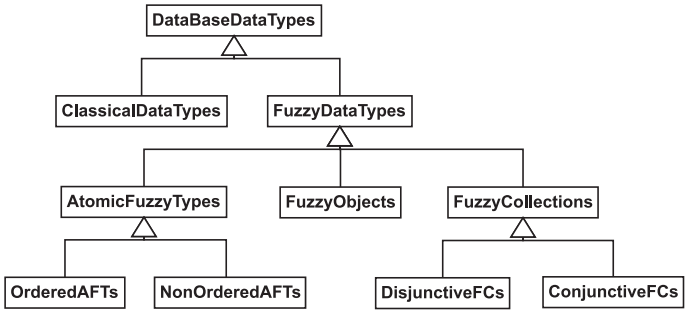


Figura A.1: Jerarquía de tipos del modelo OR Difuso

En la jerarquía podemos distinguir los siguientes tipos para el manejo de datos difusos:

- *FuzzyDataTypes (FDT).*

  Es una abstracción de todos los tipos de datos difusos soportados. Puede considerarse como un interfaz que declara métodos generales comunes que deben implementarse en los subtipos, como por ejemplo el método de instancia FEQ.

- *AtomicFuzzyTypes (AFT).*

  Declaran e implementan comportamiento común para una categoría especial de FDTs que son una extensión difusa de escalares atómicos y datos numéricos.

- *OrderedAFTs (OAFT).*

  Proporcionan estructura y comportamiento a datos atómicos difusos representados como una distribución de posibilidad definida en un dominio ordenado. Debido a que el dominio es ordenado, también define otros operadores basados en relaciones de orden (*mayor que* difuso *FGT*, *menor que* difuso *FLT*, etc...).

- *NonOrderedAFTs (NAFT).*

  Proporciona estructura y comportamiento a dominios de escalares difusos que no tienen una relación de orden.

- *FuzzyCollections (FC).*

  Generaliza el concepto de colección objeto-relacional y proporciona un comportamiento abstracto común a los subtipos. El grado difuso se refleja como la pertenencia de los elementos a la colección, sean estos difusos o no.

- *DisjunctiveFuzzyCollections (DFC).*

  Son colecciones difusas con semántica disyuntiva.

- *ConjunctiveFuzzyCollections (CFC).*

  Son colecciones difusas con semántica conjuntiva.

- *FuzzyObjects (FO).*

  Proporcionan al usuario una manera general de tratar con objetos difusos complejos. Los subtipos de FO pueden tener cualquier estructura, y sus atributos pueden ser de cualquier tipo definido en la base de datos.

El modelo define la comparación de FOs, para obtener su grado de semejanza y finalmente muestra un ejemplo de implementación sobre una base de datos *Oracle 9.2*. Con posterioridad, en [Barranco et al., 2008b], se define la implementación de este modelo sobre una base de datos objeto-relacional.

En [Barranco et al., 2005a] se describe la posible aplicación de un sistema objeto-relacional difuso a la gestión de propiedades inmobiliarias. Utilizando como base esta última implementación del modelo, se ha desarrollado una aplicación comercial que se describe en [Barranco et al., 2009a]. En este sistema, se integra la base de datos objeto-relacional difusa, con una aplicación web de búsqueda inmobiliaria. La aplicación web, proporciona interfaces de consulta adaptados, para la realización de consultas flexibles.

El problema de la implantación de los modelos de bases de datos difusos en modelos comerciales, es la falta de eficiencia. Dado que los tipos de datos que manejan son complejos y no nativos al sistema, no existen métodos para optimizar su uso. Para superar este inconveniente, en [Barranco et al., 2008a, Barranco et al., 2009b] se presentan mecanismos de indexación para optimizar consultas sobre tipos de datos numéricos difusos.

A.4. Ontologías

El término ontología se ha utilizado de muchas formas distintas y en dominios muy diversos. En el ámbito de la inteligencia artificial las ontologías fueron introducidas por Gruber y se definen como *"una especificación explícita y formal sobre una conceptualización compartida"* [Gruber, 1993]. La interpretación de esta definición, es que las ontologías definen conceptos y relaciones de algún dominio, de forma compartida y consensuada. Esta conceptualización debe ser representada de una manera formal, legible y utilizable mediante un procesamiento automático.

Las ontologías no sólo permiten la estructuración del conocimiento, sino que también permiten realizar un razonamiento sobre las afirmaciones que modelan. Mediante el uso de razonadores, se puede validar una ontología o realizar tareas tales como clasificación de instancias y clases. Dependiendo de la complejidad de la ontología será computacionalmente posible, o no, realizar alguna de las operaciones posibles.

Existen diferentes tipos de ontologías según el tipo de conocimiento que modelan. Las iremos comentando comenzando de más específica a más general.

- **Ontologías de Aplicación:** Son las ontologías mas específicas. Los conceptos de estas ontologías suelen corresponderse con los roles desempeñados por las entidades de dominio cuando se realiza una cierta actividad.
- **Ontología de Tarea:** Describe el vocabulario asociado a una actividad o tarea genérica especializando una ontología de alto nivel.
- **Ontología de Dominio:** Describe el vocabulario asociado a un dominio genérico, especializando los conceptos introducidos en la ontología de alto nivel.
- **Ontología de Alto Nivel:** Describen conceptos muy generales como espacio, tiempo, eventos, que son independientes de un problema o dominio particular. Lo más razonable es tener ontologías de alto nivel comunes para grandes comunidades de usuarios.

Por lo general lo más común es trabajar con ontologías de dominio, que describan un dominio de aplicación concreto.

A.4.1. La Web Semántica y sus Tecnologías

La Web Semántica tal y como fue concebida en [Berners-Lee et al., 2001], supone un cambio de paradigma con respecto al uso actual que se está haciendo de la web. En la web clásica la búsqueda de contenidos está orientada a la recuperación de datos atendiendo al contenido sintáctico de los documentos. La recuperación de información se realiza en base a términos del lenguaje y su co-ocurrencia dentro de un mismo documento. Esta aproximación sintáctica puede generar ciertos problemas, dado que la información contenida o referenciada por un término puede ser ambigua. Esto es debido a la ambigüedad inherente al lenguaje natural, donde distintos conceptos (en absoluto relacionados), pueden estar representados por un mismo término sintáctico.

En oposición a la aproximación sintáctica empleada actualmente, la Web Semántica propone un cambio de paradigma, en el cual las búsquedas no van a estar guiadas por la sintáctica, sino por la semántica, realizando búsquedas basadas en conceptos y relaciones entre conceptos. Esta aproximación es mucho más compleja y requiere de estructuras adicionales que permitan almacenar y gestionar el conocimiento. En el contexto de la Web Semántica el conocimiento se estructura y define por medio de ontologías.

En el ámbito de la web, las ontologías tratan acerca de la descripción precisa de la información presente en la web y de las relaciones que se establecen entre dicha información. La representación de ontologías para la web, viene dada por una serie de estándares definidos por el W3C. El World Wide Web Consortium (W3C) es un consorcio internacional compuesto por organizaciones e investigadores, cuyo objetivo es desarrollar estándares para la web. En el caso de representación de ontologías los principales estándares son RDF, RDFS y OWL.

A.4.2. Lenguajes de Representación de Ontologías

RDF

Resource Description Framework (RDF) es un lenguaje que permite representar información acerca de recursos en la web. Está especialmente indicado para la representación de metadatos acerca de recursos web, sin embargo, RDF también puede usarse para representar información sobre cosas que pueden identificarse en la web, aunque no puedan obtenerse de ella directamente. Un ejemplo de este comportamiento puede ser la información sobre productos que se encuentran disponibles en una tienda on-line.

RDF se emplea en situaciones en las que la información debe ser procesada por aplicaciones, en lugar de simplemente ser mostrada a los usuarios. RDF proporciona un entorno común que permite tanto expresar esta información, como intercambiarla entre aplicaciones sin perder el significado original.

RDF se basa en la idea de localizar objetos utilizando identificadores como los usados en la web (Uniform Resource Identifiers o URIs) y describiendo los recursos en términos de propiedades simples y valores de propiedades. Esto permite representar afirmaciones simples acerca de recursos en forma de un grafo de nodos y arcos, representando los recursos, sus propiedades y los valores de éstas.

La unidad básica de representación en RDF es la sentencia, una sentencia identifica un elemento (Sujeto), una propiedad de éste (Predicado), y el valor para dicha propiedad (Objeto). El valor de una propiedad puede ser un valor numérico, una cadena de caracteres, otro objeto, o incluso otra sentencia.

Una representación alternativa a los grafos para las sentencias RDF es la basada en triples. Un triple no es más que el conjunto de URIs que identifican al Sujeto, Predicado y Objeto de una sentencia.

RDFS

RDFS o RDF Schema es una extensión semántica de RDF. Es un lenguaje rudimentario para la definición de ontologías que proporciona los elementos básicos para la descripción

de vocabularios.

La capacidad de representación de RDFS es muy limitada y es por esto que está en desuso, habiéndose impuesto claramente OWL como lenguaje estándar de definición de ontologías en la web.

OWL

Web Ontology Language (OWL) [OWL, 2004] es una recomendación del W3C para el procesamiento de información en la web. OWL está construido sobre RDF, y por tanto al igual que RDF está escrito en XML. OWL se diseñó con el objetivo de ser interpretado por máquinas y no para ser leído por personas. En este contexto, una ontología difiere de un esquema XML (XML Schema), ya que mientras una ontología es una representación del conocimiento, un esquema XML no es más que una definición de la estructura sintáctica de un documento que se emplea para el intercambio de datos.

Las ontologías OWL pueden estar escritas en diferentes sublenguajes, cada uno de ellos con una capacidad de representación diferente permitiendo diferente complejidad.

- **OWL Full**
  Proporciona la máxima expresividad en términos de representación, pero no garantiza computacionalmente la resolución de las operaciones de razonamiento.

  En OWL Full las clases pueden tratarse simultáneamente como una colección de individuos y como individuos en sí mismas. Las propiedades de datos pueden ser inversas funcionales.

- **OWL DL**
  Proporciona la máxima expresividad posible para no perder capacidad computacional, por lo tanto se garantiza que todas las operaciones podrán realizarse y que los razonamientos serán decidibles (se realizan en tiempo finito).

  OWL DL incluye todos los elementos de OWL Full con excepción de aquellos que definen tipos separables (una clase no puede también ser considerada un individuo). OWL DL recibe esta denominación por su correspondencia con la Lógica Descriptiva (Description Logic), un campo de investigación que estudia un fragmento concreto decidible de la lógica de primer orden.

- **OWL Lite**
  Soporta un subconjunto de los elementos disponibles en OWL DL, tales como las jerarquías de clasificación y restricciones simples. Por ejemplo, sólo es posible expresar valores de cardinalidad 0 o 1.

Cada uno de estos sublenguajes extiende a su predecesor, en cuanto a capacidad de representación y a las conclusiones válidas que pueden expresarse.

Desde Octubre de 2009, existe una versión de OWL conocida como OWL 2. OWL es compatible con OWL 2, es decir, todo documento escrito en OWL es un documento OWL 2 válido. OWL 2 proporciona facilidades de escritura para algunas construcciones, la funcionalidad es la misma, pero la forma de describirlo es más sencilla. También proporciona

nuevos elementos para la definición de propiedades, capacidades extendidas para los tipos de datos, anotaciones extendidas y perfiles del lenguaje.

Los perfiles de OWL 2 son subconjuntos del lenguaje que son especialmente indicados para un tipo concreto de uso.

- **OWL 2 EL** es el conjunto máximo del lenguaje para el cual que puede asegurar que el razonamiento y la consulta, en el peor de los casos se resuelven en tiempo polinomial.
- **OWL 2 QL** es el conjunto para el que razonamiento y consulta, en el peor de los casos, se resuelve en tiempo logarítmico (como en las bases de datos).
- **OWL 2 RL** permite razonamiento en tiempo polinomial para detectar consistencia, clasificación, y comprobación de instancias. Usa tecnología basada en reglas, y está relacionado con la teoría de los programas de lógica descriptiva (DLP).

En general OWL 2 delimita mejor el tipo de elementos que se pueden usar, según la tarea que se vaya a desarrollar. Además, proporciona nuevos elementos, y formas más simples de usar elementos ya existentes.

OWL y RDF

RDF y OWL son los pilares sobre los que se está comenzando a construir la Web Semántica. Mediante OWL podemos definir las ontologías que van a determinar la representación del conocimiento del dominio que queremos modelar, pero es mediante RDF que vamos a definir las instancias que han de poblar y dar sentido a la ontología OWL. Mientras OWL representa la estructura que han de tener los datos y la semántica de estos, RDF representa a los datos en sí.

A.5. Introducción a WordNet

WordNet [Fellbaum, 1998] es una base de datos léxica de gran tamaño para el idioma Inglés, cuyo desarrollo se inició en 1985 en el Laboratorio de Ciencia Cognitiva de la Universidad de Princeton, bajo la dirección del Catedrático en Psicología George A. Miller. WordNet agrupa nombres, verbos, adjetivos y adverbios en conjuntos de sinónimos llamados *synsets* expresando un concepto determinado. Los *synsets* están conectados a través de relaciones conceptuales, semánticas y léxicas, estableciendo una red. Dicha red puede ser consultada a través de una herramienta para obtener palabras y conceptos relacionados. La estructura de WordNet lo convierte en una herramienta muy útil para el procesamiento de lenguaje natural y la lingüística computacional.

El sistema WordNet consiste en un conjunto de archivos lexicográficos, el código para convertirlos en una base de datos, y rutinas e interfaces de búsqueda para mostrar la información de la base de datos. Los archivos lexicográficos organizan nombres, verbos, adjetivos y adverbios en grupos de sinónimos y describen las relaciones entre dichos grupos.

Para poder sacar provecho de esta herramienta es muy importante conocer la forma en que el conocimiento está organizado en la base de datos.

La información en WordNet está organizada alrededor de agrupamientos lógicos llamados *synsets*. Cada *synset* es una lista de palabras individuales o colocaciones terminológicas (por ejemplo, "fountain pen" , "take in"), y apuntadores que describen las relaciones entre un *synset* y el resto. Una palabra puede aparecer en mas de un *synset* y en mas de una categoría gramatical. Las palabras de un *synset* están agrupadas de tal forma que son intercambiables en ciertos contextos.

Las relaciones modeladas por los apuntadores pueden ser de dos tipos: léxicas y semánticas. Las relaciones léxicas se mantienen entre las formas de las palabras y las semánticas entre los significados. Estas relaciones incluyen (aunque existen mas) hiperonimia/hiponimia, antonimia, implicación y meronimia/holonimia.

Los nombres y verbos se organizan en jerarquías basadas en relaciones de hiperonimia/hiponimia entre *synsets*. Existen otros apuntadores para indicar diferentes relaciones. Los adjetivos se organizan en grupos conteniendo *synsets* cabecera y *synsets* satélite, cada uno de los cuales representa un concepto con significado similar al representado por la *synset* cabecera. Una forma de pensar acerca del agrupamiento de adjetivos, es visualizar una rueda, en la que *synset* cabecera actúa como centro y los *synsets* satélite son los radios. Dos o mas ruedas estarían conectadas lógicamente a través de una relación de antonimia, que siguiendo el símil empleado, podría verse como un eje entre las ruedas.

Los adjetivos terminológicos (pertainyms), son adjetivos relacionales y no se ajustan a la estructura descrita. Estos adjetivos no tienen antónimos, por lo que el *synset* para estos, en la mayoría de ocasiones únicamente contiene la palabra o la colocación y un apuntador léxico al adjetivo al que pertenece. Los adjetivos participios tienen apuntadores léxicos a los verbos de los que se derivan.

Los adverbios en ocasiones se derivan de los adjetivos, y algunas veces tienen antónimos, por lo tanto, el *synset* para un adverbio suele contener un apuntador léxico al adjetivo del cual se deriva.

Cuando realizamos una consulta a WordNet, este nos ofrece la información agrupada

dependiendo de la categoría gramatical. Por cada categoría, ofrece un conjunto de sentidos para la palabra. Cada sentido corresponde a un *synset*, y tiene asociada una definición y una o más frases de ejemplo. Los distintos sentidos de una palabra para una determinada categoría gramatical están ordenados por la frecuencia de uso de dicho sentido. Es decir, cuando consultamos WordNet, obtendremos primero los sentidos mas utilizados de la palabra, esta característica es muy importante en el proceso de desambiguación de términos, como se ha visto en esta memoria.

A.5.1. Términos y Conceptos Básicos en WordNet

Mucha de la terminología empleada en WordNet es propia del sistema, e incluso otros términos generales tienen un significado específico en el contexto de WordNet. Es por esto, que se hace necesario hacer un breve repaso de la terminología básica y de los conceptos asociados a ella.

En las siguientes definiciones incluiremos la versión original del término entre paréntesis, con el objetivo de facilitar su reconocimiento.

- **Forma base (base form)** - La forma base de una palabra o colocación terminológica, es aquella forma básica sobre la que se añaden inflexiones.
- **Synset - Conjunto de sinónimos**. Conjunto de palabras que son intercambiables en algunos contextos sin cambiar el valor de verdad de la preposición en la que se encuentran.
- **Cluster de adjetivos (adjective cluster)** - Un grupo de *synsets* adjetivos que están organizados alrededor de pares o triples de antónimos. Un cluster de adjetivos contiene dos o mas *synsets* cabecera, que representan conceptos antónimos. Cada *synset* cabecera tiene uno o mas *synsets* satélite.
- **Atributo (attribute)** - Un nombre para el que los adjetivos expresan valores. El nombre *peso* es un atributo, para el que los adjetivos *ligero* y *pesado* expresan valores.
- **Colocación terminológica (collocation)** - Una colocación terminológica en WordNet es una cadena con dos o mas palabras conectadas por espacios en blanco o guiones. En la base de datos los espacios se representan como caracteres de subrayado (_).
- **Coordinados (coordinate)** - Términos coordinados son aquellos nombres o verbos que tienen el mismo hiperónimo.
- **Apuntador entre clusters (cross-cluster pointer)** - Un apuntador semántico de un cluster de adjetivos a otro.
- **Formas relacionadas derivacionalmente (derivationally related forms)** - Términos en diferentes categorías sintácticas que tienen la misma forma raíz y están semánticamente relacionados.

- **Dominio (domain)** - Una clasificación temática a la cual se ha enlazado un *synset* mediante un apuntador CATEGORY, REGION o USAGE.
- **Termino de dominio (domain term)** - Un *synset* perteneciente a la clase de un determinado tema. Un término del dominio se identifica adicionalmente como término de categoría (CATEGORY_TERM), término de región (REGION_TERM) o término de uso (USAGE_TERM).
- **Implicación (entailment)** - Un verbo X implica a Y, si X no puede realizarse a menos que se realice o se haya realizado Y.
- **Lista de excepciones (exception list)** - Transformaciones morfológicas de palabras que no son regulares, y por tanto no pueden procesarse de forma algorítmica.
- **Grupo (group)** - Sentidos de verbos similares en significado y que han sido agrupados manualmente.
- **Glosa o Definición (gloss)** - Cada *synset* contiene una glosa que consiste en una definición y frases de ejemplo opcionales.
- **Synset cabecera (head synset)** - *Synset* es un cluster de adjetivos, el cual contiene al menos una palabra que posee un antónimo directo.
- **Holónimo (holonym)** - El nombre del todo al que hacen referencia los merónimos. Y es un holónimo de X, si X es parte de Y.
- **Hiperónimo (hypernym)** - El término general empleado para designar una clase completa de instancias específicas. Y es un hiperónimo de X, si X es algún tipo de Y.
- **Hipónimo (hyponym)** - El término específico usado para designar un miembro de una clase. X es un hipónimo de Y, si X es algún tipo de Y. Un ejemplo bastante sencillo es la relación *gato-animal*, donde *gato* es hipónimo de *animal*, y a su vez *animal* es hiperónimo de *gato*.
- **Antónimos directos (direct antonyms)** - Par de palabras entre las que existe un enlace asociativo debido a frecuente coocurrencia. En clusters de adjetivos los antónimos directos aparecen únicamente en los *synsets* cabecera.
- **Antónimo indirecto (indirect antonym)** - Un adjetivo en un *synset* satélite que no tiene un antónimo directo, tiene un antónimos indirectos a través del antónimo directo del *synset* cabecera.
- **Instancia (instance)** - Un nombre propio cuyo referente es único (en contraposición a los nombres que hacen referencia a clases). Es una forma específica de hipónimo.
- **Lema (lemma)** - Texto ASCII en minúscula que aparece en los ficheros de índices de la base de datos de WordNet. Por lo general se corresponde con la forma base de una palabra o de una colocación terminológica.

- **Apuntador léxico (lexical pointer)** - Un apuntador léxico indica una relación entre palabras de los *synsets* (formas de palabras).

- **Fichero lexicográfico (lexicographer file)** - Ficheros conteniendo los datos de los *synsets* de WordNet, creados por expertos.

- **Identificador lexicográfico (lex id)** - Un número decimal, que añadido a un lema, identifica el sentido de forma unívoca en un fichero lexicográfico.

- **Monosémico (monosemous)** - Que tiene un único sentido en una categoría sintáctica.

- **Polisémico (polysemous)** - Que tiene más de un sentido en una categoría sintáctica.

- **Merónimo (meronym)** - El nombre de un constituyente de *parte de*, la sustancia de, el miembro de algo. X es un merónimo de Y, si X es parte de Y.

- **Categoría gramatical (part of speech - POS)** - WordNet define las siguientes categorías gramaticales: nombre, verbo, adjetivo o adverbio. Es equivalente a categoría sintáctica.

- **Adjetivo Participativo (participial adjective)** - Un adjetivo que se deriva de un verbo.

- **Adjetivo Terminológico (pertainym)** - Un adjetivo relacional. Los adjetivos de este tipo suelen definirse con frases tales como *'de o perteneciente a'* y no tienen antónimos. Un adjetivo terminológico puede apuntar a un nombre o a otro adjetivo terminológico.

- **Contador polisémico (polysemy count)** - Número de sentidos de una palabra, en una categoría sintáctica en WordNet.

- **Postnominal** - Un adjetivo postnominal aparece únicamente, justo a continuación del nombre que modifica.

- **Predicativo (predicative)** - Un adjetivo que puede usarse únicamente en posiciones de predicado. Si X es un adjetivo predicativo, sólo puede usarse en frases de tipo *'es X'* y nunca antes del nombre.

- **Prenominal** - Un adjetivo que únicamente puede aparecer antes del nombre que modifica.

- **Synset satélite (satellite synset)** - El *synset* en un cluster de adjetivos, que representa un concepto similar en significado al concepto representado por el *synset* cabecera.

- **Concordancia semántica (semantic concordance)** - Un conjunto de textos y un diccionario léxico (por ejemplo WordNet), combinados de tal manera que cada sustantivo en el texto está enlazado con su sentido apropiado en el diccionario a través de una etiqueta semántica.

- **Apuntador semántico (semantic pointer)** - Un apuntador semántico indica una relación entre *synsets* (conceptos).

- **Sentido (sense)** - Un significado de una palabra en WordNet. Cada sentido de una palabra se encuentra en un *synset* diferente.

- **Clave de sentido (sense key)** - Información necesaria para encontrar un sentido en la base de datos de WordNet. Una clave de sentido combina un campo lema y códigos para el tipo de *synset*, identificador lexicográfico, número de fichero lexicográfico y la información sobre la cabecera del *synset* satélite, en caso de que fuese necesario.

- **Etiqueta Semántica (semantic tag)** - Un apuntador desde una palabra en un fichero de texto a un sentido específico de dicha palabra en la base de datos de WordNet. Una etiqueta semántica en una concordancia semántica se representa por una clave de sentido.

- **Subordinado (subordinate)** - Equivalente a hipónimo.

- **Superordinado (superordinate)** - Equivalente a hiperónimo.

- **Tropónimo (troponym)** - Un verbo que denota la forma específica de otro verbo. Marchar es un tropónimo de andar. X es tropónimo de Y, si hacer X es hacer Y de una determinada manera.

- **Principio único (unique beginner)** - Un *synset* nominal sin superordinado.

A.5.2. Análisis Morfológico

Aunque por lo general WordNet únicamente almacena formas base, las búsquedas pueden realizarse sobre formas declinadas. Para este tipo de búsquedas, WordNet emplea *Morphy*, un conjunto funciones morfológicas que se aplican sobre la cadena de búsqueda, para generar una forma que aparezca en WordNet.

El análisis morfológico en WordNet utiliza dos tipos de procesos para convertir la cadena pasada como parámetro en una que pueda encontrarse en la base de datos. El primero consiste en una lista de terminaciones, basadas en categorías sintácticas, que pueden eliminarse de las palabras individuales para transformarlas en las formas apropiadas. El segundo consiste en una serie de ficheros con una lista de excepciones, una por cada categoría sintáctica, en los que se encuentran formas declinadas no usuales. Cuando se recibe un cadena a buscar, primero comprueba si la palabra aparece en los ficheros de excepciones, en caso afirmativo, se sustituye por la forma base correspondiente, y en caso negativo, se procede a aplicar las reglas para eliminar terminaciones.

Reglas de eliminación de terminaciones

La tabla de la figura A.1 muestra las reglas de eliminación de terminaciones empleada por *Morphy*. Si una palabra termina con alguno de estos sufijos, se elimina el sufijo correspondiente, y se añade una nueva terminación. Una vez obtenida la forma base, ya es posible buscar en WordNet. Es importante señalar que no existen reglas aplicables a adverbios.

Tabla A.1: Reglas de eliminación de terminaciones en *Morphy*

POS	**Suffix**	**Ending**
NOUN	"s"	""
NOUN	"ses"	"s"
NOUN	"xes"	"x"
NOUN	"zes"	"z"
NOUN	"ches"	"ch"
NOUN	"shes"	"sh"
NOUN	"men"	"man"
NOUN	"ies"	"y"
VERB	"s"	""
VERB	"ies"	"y"
VERB	"es"	"e"
VERB	"es"	""
VERB	"ed"	"e"
VERB	"ed"	""
VERB	"ing"	"e"
VERB	"ing"	""
ADJ	"er"	""
ADJ	"est"	""
ADJ	"er"	"e"
ADJ	"est"	"e"

Listas de Excepciones

Existe un fichero con una lista de excepciones para cada categoría sintáctica. Las listas de excepciones contienen transformaciones morfológicas no regulares, y que por tanto no pueden ser procesadas mediante un algoritmo. Cada línea de una lista de excepciones contiene una forma declinada y una lista de formas base a las que se corresponde. Los ficheros están ordenados alfabéticamente y el tipo de búsqueda empleado para buscar en las lista es la busqueda binaria.

Por lo general las palabras sueltas son fáciles de procesar, primero se buscan en WordNet y si no se encuentran resultados, se prueba a buscar en las listas de excepciones tras aplicar

la eliminación de terminaciones a la palabra. Dado que puede existir más de una forma base para una determinada declinación, sucesivas llamadas a *Morphy* nos irán ofreciendo esas posibilidades.

En cuanto al procesamiento de colocaciones sintácticas, el procesamiento para obtener la forma base es más complejo. Por lo general se procede a procesar las palabras que componen la colocación de forma individual. Esta aproximación suele funcionar con nombres, pero existen excepciones. Estas excepciones serán incluidas en las listas de excepciones apropiadas. Las colocaciones sintácticas que contienen preposiciones son más difíciles de tratar. Al igual que con las palabras individuales se intenta buscar una coincidencia en las listas de excepciones, en caso de no encontrarla, *Morphy* determina si existe una preposición. En caso afirmativo, supone que la primera palabra es un verbo y que la última es un nombre, para estas palabras obtiene la forma base correspondiente y deja las palabras intermedias tal y como estaban. En caso de que no se determine la aparición de una preposición, se procederá a buscar la forma base de cada una de las palabras que componen la colocación sintáctica.

Algunas palabras en inglés aparecen unidas mediante guiones (hyphens '-'), esto no depende de regla alguna y suelen utilizarse o no a conveniencia. En este caso, si la forma a buscar no está en WordNet se interpretan los guiones como espacios en blanco y se procede de igual modo que con las colocaciones sintácticas.

Antes de acabar conviene comentar algunas particularidades de *Morphy*. Además de excepciones, *Morphy* también almacena algunas abreviaturas que son de uso habitual. Por otra parte, realiza un tratamiento individualizado de los nombres con terminación 'ful'. *Morphy* procesa estos nombres tomando la parte que precede al sufijo 'ful', la procesa, y vuelve a añadirle el sufijo 'ful'

A.5.3. Relaciones empleadas en WordNet

WordNet, aparte de almacenar definiciones sobre términos y organizarlos en conjuntos de sinónimos, almacena relaciones semánticas entre términos. La Tabla A.2 muestra las principales relaciones semánticas modeladas en WordNet y una breve descripción de ellas.

Tabla A.2: Relaciones modeladas en WordNet

Relación	**Descripción**
ANTONYM	Antónimo, relaciona *synsets* con significados opuestos
ENTAILMENT	Un verbo implica a otro si depende de este para su realización
ENTAILED_BY	Un verbo es implicado por otro que depende de él
HYPERNYM	Hiperónimo, relaciona términos con otros que los contienen
HYPONYM	Hipónimo, relaciona un término con aquellos que contiene
SEE_ALSO	Enlaza términos relacionados
SIMILAR_TO	Indica cierta similitud entre los conceptos

Apéndice B

Herramientas utilizadas

A lo largo del desarrollo de la presente Tesis, se han utilizado diferentes herramientas software. Algunas de las herramientas utilizadas ya existían y otras han sido expresamente desarrolladas para la ocasión. En general, en este apartado hablaremos de las herramientas que hemos desarrollado, y aquellas que hemos modificado para poder integrarlas en nuestra propuesta.

Distinguiremos entre las herramientas empleadas para el almacenaje de ontologías y las empleadas para implementar la metodología de extracción de semántica a partir de texto.

B.1. Herramienta de Transformación de Ontologías a Bases de Datos

Para la transformación de ontologías a bases de datos, hemos desarrollado *OWL2SQL*, un programa Java que realiza la transformación de un fichero OWL, a un fichero de texto que contiene sentencias SQL para la creación del esquema de base de datos.

El programa presenta una aplicación de consola, a la que se pasa como parámetro un fichero OWL y un fichero de salida. La aplicación lee el documento OWL y lo procesa, y escribe las sentencias SQL en el fichero de salida. El procesamiento de la ontología se realiza usando la API para procesamiento de ontologías Jena (versión 2.6.2). Una vez leído el documento, se generan las sentencias SQL para la creación del esquema de almacenamiento de la ontología, y las necesarias para insertar la ontología de inicio. Posteriormente se aplican los algoritmos que se vieron en la Sección 3.3. La forma de proceder es, mantener en memoria una representación del esquema de la base de datos, incluyendo las dos estructuras que se definieron (esquema de almacenamiento de la ontología y esquema de almacenamiento de instancias). El esquema se va modificando con las nuevas clases y propiedades leídas de la ontología. Una vez se ha terminado el proceso, se toma el esquema en memoria y se generan las sentencias SQL necesarias para crearlo. Todas las sentencias se escriben en el orden adecuado de ejecución en el fichero de texto de salida.

El script generado se ejecuta en la base de datos, se crean todas las estructuras y se rellenan con las tuplas correspondientes.

B.2. Implementación de la Metodología

Cada una de las etapas de la metodología utiliza diversas herramientas para su implementación. A continuación describiremos el uso de las diferentes herramientas e identificaremos las etapas de la metodología con la que se corresponden.

El uso de las diferentes herramientas siempre ha ido ligado a una etapa muy concreta de la implementación de la metodología vista en la Sección 4.1. En este apéndice haremos un repaso a las distintas herramientas utilizadas, explicando en que etapas se emplean y su funcionamiento básico.

B.3. Herramientas para el Preprocesamiento Sintáctico

En todos los procesos de minería de datos y de textos, la etapa de preprocesamiento de los datos es un elemento muy importante. En nuestro caso, los conjuntos de datos van a provenir de tablas de una base de datos Objeto-Relacional. Por esto, hemos diseñado una herramienta genérica que nos permita realizar un proceso de limpieza de datos adaptado a nuestras necesidades.

En esta sección se procederá a detallar la descripción de la herramienta genérica de filtrado de datos, que permite realizar un proceso de limpieza selectiva sobre un conjunto de datos almacenados en una base de datos, con el fin de obtener un conjunto de datos sobre el que aplicar algoritmos de minería de datos y de textos.

B.3.1. Herramienta de Filtrado de Datos

La herramienta desarrollada cuenta con un interfaz sencillo e intuitivo, que permite a un usuario sin entrenamiento previo en el manejo de la misma, la construcción de datasets a partir de un esquema de base de datos previo.

La implementación de la herramienta está realizada en Java usando JDBC para poder adaptarla a la mayor cantidad posible de SGBDs. Adicionalmente, la herramienta proporciona la posibilidad de elegir entre el procesamiento completo, o de realizar llamadas a métodos nativos que permitan la ejecución de dicho procesamiento de forma interna en la base de datos.

Lo primero es establecer una conexión con la base de datos. Estas conexiones podrán realizarse introduciendo los datos de conexión y ofreciendo la posibilidad de hacerlas persistentes (evitando así tener que definirlas cada vez). Cuando intentamos conectar a través de una conexión realizada previamente, se hará de forma automática, excepto si no se marcó para recordar el password, en cuyo caso presentaría el nombre de usuario y esperaría la introducción del password. Se podrá guardar la configuración de varias conexiones distintas, pero sólo se podrá mantener una activa. Las conexiones persistentes podrán ser modificadas y borradas. La datos de configuración de las conexiones persistentes serán almacenados en XML, con la finalidad de dotarlos de cierta semántica y hacerlos portables, permitiendo importarlos y exportarlos.

Una vez conectados a la base de datos, podemos comenzar a seleccionar datos. Los mecanismos de selección horizontales nos permitirán seleccionar tuplas a lo largo de todo

el esquema bajo una condición impuesta por el usuario sobre uno o más campos. Los mecanismos de selección verticales nos permitirán realizar proyecciones sobre el conjunto de datos y seleccionar aquellas tablas y campos que contengan los datos que consideramos necesarios. Se permite la introducción de condiciones complejas en modo texto.

Una vez realizada la selección de datos, debemos identificar las operaciones de filtrado que vamos a realizar, para ello seleccionamos los campos que vayamos a procesar y para cada uno de ellos indicamos los procesos de filtrado a realizar y el orden, de forma independiente. Como opción adicional se permitirá definir el mismo tipo de filtrado sobre grupos de campos (evitando así definir el mismo proceso para cada uno de los datos).

Tras la realización de las operaciones de discretización y filtrado de textos, obtenemos un conjunto de metadatos que nos permitirán la posterior generación de datasets relacionales y transaccionales.

El conjunto de operaciones realizadas sobre un conjunto de datos en cada etapa quedarán almacenadas en un fichero de configuración XML para poder conocer el procesamiento que ha dado origen a los datos actuales. Este fichero podrá ser utilizado para repetir el proceso para un conjunto de datos diferente pero que responda al mismo esquema de BD.

La herramienta dispone adicionalmente de un asistente de generación de datasets en modo offline que realizará todo el procesamiento completo de forma guiada, siguiendo los siguientes pasos:

- Conectarse a una BD
- Seleccionar tablas de un listado
- Aplicar condiciones de selección a los atributos (propagación horizontal). Realizamos este paso antes de la selección vertical por si queremos establecer condiciones sobre atributos que no vamos a incluir en el dataset resultante.
- Indicar el tipo de filtrado que se va a realizar sobre cada uno de los campos
  - Tokenización.
  - Aplicación de Diccionarios.
  - Eliminación de Stop Words.
  - Lematización (Stemming).
  - Ordenación alfabética.
  - Otros.

En caso de seleccionar más de un filtrado se indicará el orden en que se deben realizar estos

- Incluir filtrado a nivel de registros para eliminar tuplas repetidas.
- Discretizar campos numéricos y seleccionar las representaciones intermedias a generar.

- Conjunto de metadatos a generar.
- Tablas relacionales a crear.
- Tablas transaccionales a crear.

B.3.2. Descripción detallada

Veamos cada una de las etapas de forma detallada.

Tablas → Vistas - Creación de una consulta

A partir de un esquema de bases de datos y de forma gráfica, definimos una consulta mediante la cual realizamos una selección vertical y horizontal. Para la selección vertical contamos con una serie de checkboxes, al pulsarlos seleccionamos la columna para la inclusión en el dataset. Para la selección horizontal contamos con mecanismos para realizar reunión entre tablas, y para aplicar condiciones de selección según un atributo.

Tras la selección se genera una consulta SQL que va a definir una vista.

Vistas → Metadatos - Aplicación de filtros y discretización

Campos de Texto

Una vez obtenida la vista, es posible aplicar un filtrado a los campos textuales. Este filtrado puede definirse para uno o mas campos. Los filtros a aplicar son:

- Tokenización.
- S-Stemmer (Inglés).
- Porter Stemming (Inglés).
- S-Stemmer (Español).
- Porter Stemming (Español).
- Eliminación de Stop Words (Inglés).
- Eliminación de Stop Words (Español).

El orden de aplicación de los filtros es relevante, ya que dependiendo del orden de estos los resultados pueden variar. Los textos filtrados se almacenan junto al resto de datos en la estructura del catálogo que llamamos METADATOS.

Un término puede estar compuesto por una o más palabras. El conjunto de términos de más de una palabra se determinarán mediante el uso de un diccionario. El ámbito de los diccionarios debe definirse, pudiendo utilizarse un diccionario general para toda la BD un diccionario por usuario, o un diccionario por columna.

Campos numéricos

Los campos numéricos podrán discretizarse en forma de:

- CRISP

  - Intervalos semi-automáticos (equidistantes o equiprobables). Parámetro N (número de intervalos).
  - Manual (definición mediante interfaz gráfico).

- Fuzzy
  - Distribuciones de posibilidad (semi-automático y manual)
  - Etiquetas Lingüísticas

Una vez discretizados los valores numéricos y filtrados los textos, debemos almacenar toda la información disponible en forma de Metadatos, de modo que a partir de ellos podamos generar los datasets que necesitamos.

Metadatos → Dataset Relacional/Transaccional

A partir de los Metadatos generados en el proceso de filtrado y discretización, construiremos los datasets. En el caso de los campos textuales se puede optar por reagrupar los textos, o mantenerlos como términos individuales. Dependiendo del tipo de uso que se vaya a dar a los datos será necesario generar el dataset con una ciertas características. La herramienta puede generar tanto dataset relacionales, como transaccionales. También se da la posibilidad de seleccionar el tipo de salida, a base de datos, o a fichero de texto con un formato determinado.

B.3.3. Diseño e Implementación

A continuación mostraremos de forma detallada cada uno de las tareas que puede realizar la herramienta haciendo hincapié en las decisiones tomadas durante la etapa de diseño. En esta sección también trataremos aspectos generales de la implementación que permitirán tener una visión general de la arquitectura de la aplicación. Principalmente nos centraremos en la descripción de aquellos elementos que hemos creado como parte del catálogo de base de datos y que permitirán realizar de forma genérica ciertos procesos.

Conexiones de Base de Datos

La descripción de las conexiones de bases de datos creadas por la aplicación se almacena en un fichero XML, siguiendo la misma estructura de fichero que la empleada por herramientas como SQLDeveloper de Oracle™, con la finalidad de hacer posible la importación/exportación de la información sobre conexiones. El fichero de configuración con los datos de las conexiones se llama `IDEConnections.xml`, y se encuentra en el directorio `/conf` de la aplicación.

A continuación mostramos un ejemplo del fichero de conexión para una conexión Oracle™llamada *Ejemplo*, para conectar a la cuenta `ejemplo1`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE connections>
<connections>
```

```
    <connection>
        <ConnectionName>Ejemplo</ConnectionName>
        <OracleConnectionType>Basic</OracleConnectionType>
        <JDBC_PORT>1521</JDBC_PORT>
        <HOSTNAME>localhost</HOSTNAME>
        <SID>SERVICE_IDENTIFIER</SID>
        <URL>jdbc:oracle:thin:@localhost:1521:SERVICE_IDENTIFIER</URL>
        <DeployPassword>false</DeployPassword>
        <user>ejemplo1</user>
        <RaptorConnectionType>Oracle</RaptorConnectionType>
        <ConnectionType>JDBC</ConnectionType>
        <JdbcDriver>oracle.jdbc.driver.OracleDriver</JdbcDriver>
        <ORACLE_JDBC_TYPE>thin</ORACLE_JDBC_TYPE>
    </connection>
</connections>
```

Cada nueva conexión se añadirá como un nuevo elemento `<connection>` dentro del elemento `<connections>`.

Información sobre transformaciones de datos

El proceso de transformación de datos desde las tablas originales de la base de datos hasta obtener el dataset final, pasa por una serie de etapas intermedias. Estas etapas intermedias pueden considerarse como hitos y permiten la generación de versiones alternativas a partir de un mismo conjunto inicial. De este modo, a partir de una selección inicial, podemos someter a los datos a diferentes procesamientos numéricos y textuales, y comparar los resultados obtenidos en cada caso.

En este sentido, podemos ver la generación de datasets como la generación de un árbol, en donde la raíz serán las tablas de la base de datos con las que estemos trabajando, y las hojas serán cada uno de los conjuntos de datos generados. Cada vez que se genera uno de los nodos intermedios de este proceso, se almacena la información relativa a como se obtuvo.

Para mantener un registro acerca de cada uno de estos nodos, hemos creado una tabla de catálogo llamada `DB2DS__VIEWS`. Esta tabla almacena el nombre de la tabla que se genera en un etapa determinada y un fichero XML que describe la forma en que se ha generado. Esto permite regenerar esa tabla (nodo) intermedio aunque el conjunto de datos de origen haya sido modificado.

A continuación mostramos el esquema de la tabla de catálogo `DB2DS__VIEWS`.

Column Name	Data Type
TABLENAME	VARCHAR2(255 BYTE)
XMLDESCRIPTION	CLOB

Figura B.1: Tabla `DB2DS__VIEWS`

Esta tabla contiene dos atributos:

- `Tablename` : Nombre de la tabla, vista o conjunto de metadatos generados.
- `XMLDescription` : Descripción en XML del proceso que se ha seguido para lograr obtener el elemento. En el caso de las vistas, será una consulta, para los metadatos, la tabla original y los filtros y discretizaciones realizadas.

Selección Horizontal y Vertical de datos

En el caso de la selección vertical no nos encontramos con excesivas dificultades, ya que basta con seleccionar las claves primarias de cada tabla que vayamos a incluir, y los campos que se hayan seleccionado previamente. La única consideración a tener en cuenta, es la que concierne a la referencias de clave externa a tablas no incluidas. Se puede optar por dos soluciones:

- *No incluir las restricciones.*
  En este caso se perdería cierta semántica, ya que podría darse el caso de que se incluyesen valores que no cumpliesen las restricciones no incluidas. Así pues nos encontramos ante un caso en el que nuevamente disponemos de varias alternativas. Si el dataset que se va a crear va a ser estático, no va a ser actualizado, puede que no tenga sentido incluir las restricciones dado que en el momento en que se creó el dataset, los datos eran consistentes. Sin embargo, si queremos que el dataset se actualice y se sincronice con una fuente de datos, si que va a ser importante y necesario conservar las restricciones.

- *Hacer que las restricciones apunten a la tabla original.*
  En este caso mantendríamos la semántica inalterada, pero sería necesario tener los privilegios necesarios para poder realizar esta operación

En nuestro caso optaremos por utilizar el modelo estático, pero como trabajo futuro parece interesante mantener las referencias y tratar de crear datasets sincronizados con las fuentes originales.

Es posible definir condiciones para campos que no han sido seleccionados para formar parte del dataset final. Sería necesario establecer una forma sencilla de realizar reuniones entre tablas, mientras tanto pueden realizarse siguiendo este método.

También es necesario incluir la posibilidad de crear una cláusula de selección de forma textual, tanto para consultas completas como para únicamente condiciones de la consulta (la selección vertical se realizaría sobre el interfaz).

En cuanto a la selección horizontal, ésta es un corte horizontal sobre el conjunto de todos los datos del esquema, seleccionando según un criterio predeterminado. En este caso es necesario establecer reuniones entre las diferentes tablas de modo que cada tupla contenga los datos referidos a una determinada entidad. Dado que el número de tablas en cada esquema puede variar y que el número de tablas puede ser muy alto parece conveniente pensar en optimizar el número de reuniones, para optimizar el rendimiento.

Filtrado de Textos

En el caso del filtrado de textos, lo que se hace es poner a disposición del usuario una serie de procedimientos que permitirán procesar los campos textuales. Todos estos procedimientos y todos los demás necesarios para el funcionamiento de la herramienta se encuentran implementados en PL/SQL en un paquete llamado `DB2DS__PKG`. Por cada campo de tipo texto se indicará un campo adicional junto a la estructura empleada para la selección, en el que se registrarán los procedimientos a aplicar y el orden en el que han de ser aplicados.

A la hora de diseñar el sistema de aplicación de filtros, hemos tenido en cuenta que éste debe ser lo más genérico posible para que permita añadir nuevos tipos de filtros una vez la herramienta ha sido finalizada. Para ello, definimos una interfaz para los métodos de filtrado, en la que todo método de filtrado implementado en PL/SQL del lado del servidor deber seguir esta estructura:

```
FUNCTION DB2DS__NOMBREFILTRO( cadena IN varchar2 ) RETURN VARCHAR2;
```

Como sugerencia y a modo de autodocumentación se recomienda que todos los elementos pertenecientes a la herramienta, ya sean estructuras de tablas o funciones y procedimientos, comiencen con el prefijo `DB2DS__`. Todas las funciones que realicen operaciones de filtrado se van a encontrar recogidas en una tabla del catálogo del sistema llamada `DB2DS__FILTERS`.

Column Name	Data Type
PROCEDURE	VARCHAR2(50 BYTE)
NAME	VARCHAR2(255 BYTE)
DESCRIPTION	VARCHAR2(500 BYTE)

Figura B.2: Tabla `DB2DS__FILTERS`

Esta tabla contiene tres atributos:

- `Procedure` : Nombre del procedimiento que implementa el filtro.
- `Name` : Nombre descriptivo del filtro, es el que se mostrará en la herramienta.
- `Description` : Breve descripción del funcionamiento del filtro.

Sólo aquellos filtros que se encuentren registrados en esta tabla del sistema podrán utilizarse en la aplicación.

Puede que algunos filtros necesiten de estructuras adicionales para contener datos (como es el caso del filtro de eliminación de stop-words), en este caso la recomendación es crear dichas tablas como tablas del sistema, con el prefijo `DB2DS__`.

Cuando se aplican los filtros sobre los datos se recogen estadísticas acerca de las modificaciones realizadas en los procesos de limpieza. Las estadísticas se recogen en dos tablas del catálogo `DB2DS__STATS` y `DB2DS__STATS_FILTERS`.

Column Name	Data Type
ID	NUMBER
ORIGINALTABLE	VARCHAR2(50 BYTE)
FILTEREDTABLE	VARCHAR2(50 BYTE)
COLUMNNAME	VARCHAR2(50 BYTE)
TOTALROWS	NUMBER
TOTALWORDS	NUMBER
UNCHANGEDWORDS	NUMBER
TOTALCHANGES	NUMBER

Column Name	Data Type
ID	NUMBER
FILTERNAME	VARCHAR2(100 BYTE)
CHANGES	NUMBER

Figura B.3: Tablas `DB2DS__STATS` y `DB2DS__STATS_FILTERS`

La tabla `DB2DS__STATS` recoge estadísticas generales sobre los cambios realizados en el texto, mientras que `DB2DS__STATS_FILTERS` ofrece detalles más específicos acerca de los cambios realizados por cada uno de los filtros.

La tabla `DB2DS__STATS` contiene los siguientes atributos:

- `ID` : Identificador de la estadística.
- `OriginalTable` : Nombre de la tabla orginal a procesar.
- `FilteredTable` : Nombre de la tabla en la que se almacenará el campo procesado.
- `ColumName` : Nombre de la columna procesada.
- `TotalRows` : Número total de filas procesadas.
- `TotalWords` : Número total de palabras proceasadas.
- `UnchangedWords` : Palabras que no han sido modificadas.
- `TotalChanges` : Número total de cambios realizados.

La tabla `DB2DS__STATS_FILTERS` hace referencia a la tabla tabla `DB2DS__STATS`, por cada entrada de estadísticas, aquí se especifican los cambios realizados por cada filtro.

- `ID` : Identificador de la estadística. Hace referencia a `DB2DS__STATS.ID`
- `FilterName` : Nombre del filtro aplicado.
- `Changes` : Número total de cambios realizados por el filtro.

Una vez aplicados los filtros se genera un conjunto de metadatos con el resultado. Este conjunto se da de alta en la tabla `DB2DS__VIEWS` junto con el archivo XML que describe el conjunto de filtros y discretizaciones empleados para obtenerlo.

Diccionarios

A la hora de procesar los campos textuales, por defecto se considera que cada palabra representa un término, pero esto no siempre es así, por ejemplo *Minería de Datos* y *Lógica Difusa*, son términos únicos formados por varias palabras. El procesamiento de este tipo de términos complejos se realiza por medio de diccionarios.

Los diccionarios recogen los distintos términos complejos, y se encargan de preservarlos mientras se procesan los datos. Estos diccionarios pueden ser definidos por los proprios usuarios, añadiendo, modificando o borrando nuevos términos. Para facilitar el proceso de generación de diccionarios, es posible crearlos a partir de otros ya definidos, o incluir nuevos términos provenientes de diferentes diccionarios.

Las estructuras definidas para el uso de diccionarios consisten en una tabla de catálogo que contiene el nombre y definición de los diferentes diccionarios definidos, y una tabla de catálogo por cada diccionario, que contiene los términos incluidos en este. Como siempre, se recomienda que las tablas de diccionarios empiecen por el prefijo `DB2DS__`.

Column Name	Data Type
DICTIONARYTABLE	VARCHAR2(50 BYTE)
NAME	VARCHAR2(255 BYTE)
DESCRIPTION	VARCHAR2(500 BYTE)

Figura B.4: Tablas `DB2DS__DICTIONARIES`

La tabla `DB2DS__DICTIONARIES` contiene los siguientes atributos:

- `DictionaryTable` : Nombre de la tabla que contiene los términos del diccionario.
- `Name` : Nombre descriptivo del diccionario, es el que se mostrará en la herramienta.
- `Description` : Breve descripción del contenido del diccionario.

Metadatos

Una vez procesados los datos y aplicados los filtros, estos se almacenan en unas tablas del sistema, con el objetivo de generar datasets con diferentes formatos a partir de los mismos datos. Para esto, necesitamos almacenar datos adicionales acerca de la información de tal forma que podamos reconstruirla a posteriori.

Las tablas que organizan los metadatos son las siguientes:

La tabla `DB2DS__OVIEWS` contiene una relación de los conjuntos que se van a generar para una determinada vista. Recordemos que ya habíamos comentado que para cada vista (entendida esta como una selección horizontal y vertical) se podían generar distintos conjuntos de metadatos, generados a partir de la aplicación de distintos filtros.

Esta tabla tiene los siguientes atributos:

- `ID_View` : Nombre de la tabla/vista a la que pertenecen los metadatos.

Column Name	Data Type
ID_VIEW	VARCHAR2(40 BYTE)
ID_TSET	VARCHAR2(40 BYTE)

Column Name	Data Type
ID_TSET	VARCHAR2(40 BYTE)
ID_COLUMN	VARCHAR2(40 BYTE)
ID_TRANSACTION	VARCHAR2(18 BYTE)
ID_ITEM	NUMBER(10,0)
POSITION	NUMBER(10,0)
WEIGHT	FLOAT

Column Name	Data Type
ID_ITEM	NUMBER(10,0)
ITEM	ANYDATA

Figura B.5: Tablas `DB2DS__OVIEWS`, `DB2DS__TSETS` y `DB2DS__OITEMS`

- `ID_TSet` : Nombre del conjunto de metadatos generados.

La tabla `DB2DS__TSET` contiene la información acerca de los metadatos generados, describe todos los elementos que pertenecen a una misma tupla. Contiene los siguientes atributos:

- `ID_TSet` : Nombre de la tabla/vista a la que pertenecen los metadatos. Hace referencia a `DB2DS__OVIEWS.ID_TSET`
- `ID_Column` : Nombre del conjunto de metadatos generados.
- `ID_Transaction` : Identificador de la fila a la que pertenece el elemento.
- `ID_Item` : Identificador del elemento. Hace referencia a `DB2DS__OITEMS.ID_ITEM`
- `Position` : Posición del elemento en caso de ser un campo con múltiples valores. En el caso de campos textuales, representa la posición del término en el campo.
- `Weight` : Peso asociado al término.

Dado que los elementos (*items*) pueden ser de diferentes tipos, los gestionamos en una tabla aparte que contiene un elemento de tipo `ANYDATA`, este tipo puede contener cualquiera de los tipos simples y complejos definidos en la base de datos.

La tabla `DB2DS__OITEMS` tiene los siguientes atributos:

- `ID_Item` : Identificador del elemento, es un número secuencial.
- `Item` : Campo `ANYDATA` que contiene el elemento.

Gracias a estas tablas, es posible reconstruir los datos, y expresarlos en diferentes formatos de salida.

Generación de datasets

Todo el procesamiento queda aplazado hasta este punto, hasta ahora lo que se ha hecho es definir el tipo de procesamiento que se iba a realizar. Es en este preciso momento, cuando vamos a realizar todas las operaciones que conducirán a la obtención del dataset. A partir de los metadatos obtenidos en las etapas anteriores construimos un dataset relacional o transaccional, según nos convenga, y efectuaremos la salida en forma de tabla en la base de datos o de fichero de texto. Los datasets generados serán utilizados por otras herramientas para realizar procesos de minería de datos y texto sobre ellos.

B.4. Herramienta para el Preprocesamiento Semántico

Para realizar esta etapa hemos creado *SemanticPreprocessor*, un programa Java que realiza todas las etapas del procesamiento semántico. Para alguna de estas etapas hemos utilizado algunas herramientas externas. Veamos las distintas tareas a realizar, y las herramientas usadas.

Partiendo de una columna de base de datos, leemos los textos, se procesan, y se crea una columna nueva en la tabla con los textos procesados.

El programa toma como parámetro un fichero de configuración, donde se indican los datos de la conexión con la base de datos, la columna a procesar, la tabla en la que se encuentra la columna, y el nombre de la columna en la que se escribirán los textos procesados. También se le pasan datos para acceder a la herramienta externa a usar, en este caso WordNet.

La herramienta externa que hemos utilizado en esta etapa es WordNet, en su versión 3.0 para sistemas UNIX. El acceso a WordNet a través de nuestro programa Java, lo hemos realizado a través de la API de la librería JWNL, en su versión 1.4-rc2 de 10 de Julio de 2008, la última hasta la fecha.

El procesamiento semántico consiste en las siguientes etapas:

- *Etiquetado de categoría gramatical (Part Of Speech Tagging)*: Tomamos el contenido textual de una fila, y se etiquetan los términos con la categoría gramatical que desempeña en la sentencia de la que se ha extraído. El etiquetado se realiza utilizando la API para etiquetado lingüístico de *The Stanford Natural Language Processing Group* [Toutanova and Manning, 2000, Toutanova et al., 2003]. Usamos la librería `stanford-postagger.jar`, concretamente la versión 3.0 de 26 de Mayo de 2010. Pasando el texto al etiquetador, este devuelve una lista de palabras con su etiqueta correspondiente. Esa lista de palabras es procesada, y adaptamos las etiquetas *Penn Treebank* a WordNet. En la Sección 4.4.2 se describe el proceso de etiquetado de forma teórica y se muestra la tabla de correspondencias entre ambos conjuntos de etiquetas. Cuando las palabras han sido etiquetadas se procede a su desambiguación.

- *Desambiguación (Word Sense Disambiguation)*: Se determina el significado del término en cuestión, ya que, una vez conocido su significado pueden determinarse sus sinónimos. En este punto, tomamos la lista de palabras etiquetadas, y le aplicamos

el algoritmo de desambiguación. Hemos implementado los siguientes algoritmos de desambiguación:

- *Sin desambiguación*: Lo incluimos para poder comparar los resultados de las aproximaciones que usan desambiguación con un caso base, y comprobar si mejoran los resultados.
- *El sentido más frecuente*: Se toma el sentido más frecuente de un término. Para este algoritmo usamos WordNet. Dado que WordNet almacena los sentidos por orden de frecuencia, es tan sencillo como devolver el primer sentido disponible.
- *Algoritmo simplificado de Lesk*: Este algoritmo se basa en el solapamiento del contexto de una palabras con las definiciones de los sentidos en WordNet. La implementación la hemos realizado siguiendo las especificaciones del algoritmo presentadas en en [Vasilescu et al., 2004].
- *Algoritmo adaptado de Lesk*: Este algoritmo es muy similar al anterior, con la salvedad de que en la desambiguación no sólo se emplea la definición de cada uno de los sentidos en WordNet, sino también las definiciones de palabras relacionadas, hipónimos, tropónimos, etc. La implementación de este algoritmo la hemos realizado según la descripción de [Banerjee and Pedersen, 2002].

- *Generación de conjuntos de sinónimos*: Una vez desambiguadas toda las palabras del atributo textual, éstas se agrupan atendiendo a su sentido. Todos los términos en un mismo grupo se consideran sinónimos.

- *Selección del representante canónico de los conjuntos de sinónimos*: Para cada conjunto de sinónimos se selecciona el término más frecuente. Dicho término sustituirá a todos los términos de su conjunto de sinónimos en el texto original.

Una vez finalizado el proceso, el texto modificado se incluye en la columna indicada en el fichero de configuración.

Para facilitar el uso de este programa, en el futuro se integrará dentro de la herramienta de preprocesamiento sintáctico presentada en la sección anterior.

B.5. Herramientas para la Generación de Formas Intermedias

TextAnalyzerTest es un programa Java que, a partir de un atributo textual en una base de datos PostgreSQL, analiza el texto, genera una representación basada en AP-Sets y almacena dicha representación en la base de datos junto al atributo original. Este programa ha sido desarrollado por los doctores Serrano Chica de la Universidad de Jaén (España) y Martínez-Folgoso de la Universidad de Camagüey (Cuba). Una descripción detallada de su funcionamiento y una definición formalizada de los Conjuntos-AP se pueden encontrar en [Martínez-Folgoso, 2008].

PostgreSQL presenta algunas limitaciones en cuanto a la representación de objetos complejos, por lo que decidimos abordar la implementación de los Conjuntos-AP en un sistema

objeto-relacional mas potente, como es Oracle^{TM}. A través de la implementación de un paquete SQL, creamos la definición de la estructura de los Conjuntos-AP, sus propiedades, métodos y algunas funciones auxiliares para poder usarlos desde programas externos. Con este paquete y los métodos de interfaz, en colaboración con el Dr. Martínez-Folgoso se modificó el programa original para soportar Oracle^{TM}.

El funcionamiento es bastante sencillo, sólo es necesario definir todos los parámetros en un fichero de configuración. El programa lee el fichero de configuración, a través de la conexión a la BD lee los datos, calcula los itemsets maximales con el soporte y confianza indicados en el fichero de configuración y genera la salida. La salida se realiza a ficheros de texto, y además se incluye la Estructura-AP en la base de datos, y se calcula la sub-AP-Estructura correspondiente para cada una de las tuplas procesadas.

Posteriormente realizamos unos pequeñas modificaciones para poder procesar términos que tuviesen asociado su identificador de *synset* en WordNet, y por tanto, su sentido. Los términos etiquetados con su sentido se generan en la etapa de preprocesamiento semántico. Una vez se integren ambas herramientas esto no será necesario, ya que se aplicarán todas las etapas del proceso de forma secuencial en memoria.

B.6. Herramientas para la Extensión Semántica

Para la extensión semántica de los Conjuntos-AP hemos desarrollado dos programas diferenciados *WordNetOntologyCreator* y *WikipediaOntologCreator*, para la extensión con WordNet y Wikipedia respectivamente. La diferenciación en dos programas es por motivos conceptuales, ya que ambos pueden integrarse en un único programa sin mucho esfuerzo.

- *WordNetOntologyCreator*: Este programa toma la definición de una Estructura-AP de dominio, y con la ayuda de WordNet la transforma en una ontología. Esta ontología de salida se enriquece con términos adicionales de consulta extraídos de WordNet. El programa tiene como entrada un fichero de configuración y devuelve una ontología OWL. Los parámetros del fichero de configuración se expresan en la forma `propiedad=valor`. Las propiedades del fichero de configuración son:
  - Localización de la Estructura-AP.
    - **apstructure.path** - Ruta al directorio de la Estructura-AP.
    - **apstructure.name** - Nombre de los ficheros que contienen la Estructura-AP.
  - Configuración de WordNet.
    - **wordnet.configuration.file** - Determina la localización del fichero de configuración necesario para utilizar WordNet desde la API.
  - Indicaciones acerca del formato de los items (recordemos que van anotados con su *synset* de WordNet).
    - **dterms.format** - Determina el formato en el que están expresados los pares itemset/sentido. Puede tener los siguientes valores:

        - ◇ **1** : Personalizado.
          En caso de que se elija el formato personalizado hay que rellenar otra propiedad.
          dterms.separator - Carácter que delimita la separación entre un término y su identificador de *synset* en WordNet.
        - ◇ **2** : Formato WordNet
    - Método a utilizar para la generación de la ontología.
        - **algorithm.wordnet.method**
            - ◇ **1** - Crea la ontología a partir de la Estructura-AP sin usar herramientas externas.
            - ◇ **2** - Crea la ontología usando la Estructura-AP y WordNet.
    - Espacio de nombres para la ontología.
        - **ontology.namespace**
    - Nombre y localización del fichero de salida con la ontología.
        - **output.path**

- *WikipediaOntologyCreator*: Este programa toma la definición de una Estructura-AP de dominio, y con la ayuda de Wikipedia la transforma en una ontología. Esta ontología de salida se enriquece con términos adicionales de consulta extraídos de WordNet. La entrada al programa consiste en un fichero de configuración, y la salida es una ontología OWL. Los parámetros del fichero de configuración se expresan en la forma `propiedad=valor`. A continuación mostramos las propiedades del fichero de configuración:

    - Configuración de acceso a Wikipedia.
        - **db.host** - Localización del servidor de bases de datos.
        - **db.name** - Nombre de la base de datos.
        - **db.user** - Usuario de la base de datos.
        - **db.password** - Password para la cuenta de base de datos.
    - Localización del grafo de categorías de Wikipedia.
        - **wikipedia.categorygraph.path** - Ruta al fichero del grafo de categorías.
        - **wikipedia.categorygraph.name** - Nombre del fichero del grafo de categorías.
    - Configuración de WordNet.
        - **wordnet.configuration.file** - Determina la localización del fichero de configuración necesario para utilizar WordNet desde la API.
    - Localización de la Estructura-AP.
        - **apstructure.path** - Ruta al directorio de la Estructura-AP.
        - **apstructure.name** - Nombre de los ficheros que contienen la Estructura-AP.

- Formato de los itemsets en la estructura.
  - **dterms.format** - Determina el formato en el que están expresados los pares itemset/sentido. Puede tener los siguientes valores:
    - **1** : Personalizado.
      En caso de que se elija el formato personalizado hay que rellenar otra propiedad.
      dterms.separator - Carácter que delimita la separación entre un término y su identificador de *synset* en WordNet.
    - **2** : Formato WordNet
- Configuración del método a usar y sus parámetros según el caso. Los diferentes parámetros que nos encontramos son:
  - **algorithm.wikipedia.method** - Se indica el tipo de método a usar. Los métodos disponibles son:
    - **1** - Se corresponde con la exploración básica.
    - **2** - Corresponde a la exploración básica resumida.
    - **3** - Exploración básica con longitud de camino.
    - **4** - Exploración básica resumida con longitud de camino.
    - **5** - Camino más corto.
    - **6** - Método taxonómico combinado.
    - **7** - Método basado en medidas.
  - **algorithm.wikipedia.n** - Se aplica para los métodos 3, 4 y 6. Es un número entero, que indica el valor de longitud para los caminos seleccionados.
  - **algorithm.wikipedia.summarytype** - Se aplica al método 4. Indica el tipo de resumen que se puede realizar. Los posibles valores son:
    - **1** - Se seleccionan las N categorías más cercanas al *lcs*.
    - **2** - Se seleccionan las N categorías más cercanas a la categoría inicial.
    - **3** - Se seleccionan las N categorías más cercanas al *lcs* y a la categoría inicial.
    - **4** - Se seleccionan las N categorías a lo largo del camino, en intervalos regulares.
  - **algorithm.wikipedia.threshold** - Se aplica al método 7, la selección mediante medidas. Es el umbral mínimo de similitud/relación entre dos categorías para que sean seleccionadas en el resultado final. Es un valor decimal.
  - **algorithm.wikipedia.measure** - Se aplica al método 7, la selección mediante medidas. Las medidas disponibles son:
    - **1** - Jaccard.
    - **2** - Sorensen.
    - **3** - Mountford.
    - **4** - Overlap.
    - **5** - Rada.

- ◇ **6** - Leacock-Chodorow.
- ◇ **7** - Wu-Palmer.
- ◇ **8** - Resnik.
- ◇ **9** - Lin.
- ◇ **10** - Jiang-Conrath.
- ◇ **11** - Coseno.

- ○ **algorithm.wikipedia.measure.scope** - Se aplica sobre el método 7. Los posibles valores y las medidas a las que se aplican son:
  - ◇ **1** - Ámbito de comparación con la categoría inicial. Se usa con todas las medidas.
  - ◇ **2** - Ámbito de comparación con la categoría actual. Se aplica a todas las medidas excepto a 5 (es aplicable, pero no tiene sentido).
  - ◇ **3** - Ámbito de comparación con la categoría acumulada. Se aplica únicamente a las medidas 1, 2, 3, 4 y 11.
- ○ **algorithm.wikipedia.measure.cosine.stem** - Determina el uso de la lematización sobre los textos a procesar. Se aplica únicamente sobre la medida 11. Los valores posible son:
  - ◇ **0** - No se aplica lematización.
  - ◇ **1** - Se aplica lematización.
- ○ **algorithm.wikipedia.measure.cosine.datasource** - Selecciona la fuente del texto a procesar. Existen dos posibilidades:
  - ◇ **1** - Se procesa el texto del título.
  - ◇ **2** - Se procesa el texto de la página completa.

- Espacio de nombres de la ontología.
  - ○ **ontology.namespace**
- Nombre y localización del fichero de salida.
  - ○ **output.path**

De cara a la integración de estos programas en un interfaz visual, deberíamos ofrecer la posibilidad de realizar un procesamiento completo en memoria, a partir de un atributo de base de datos.

B.7. Herramientas para la Evaluación de Ontologías

En el Capítulo 6, hemos planteado la evaluación de las ontologías obtenidas en tres vertientes, relevancia semántica, estructural y basada en una aplicación. Para éstas dos últimas hemos desarrollado programas que realizan la evaluación.

- **Evaluation**: Calcula valores para las propiedades estructurales de la ontología. Las propiedades utilizadas se describen en la Sección 6.3.2. Utilizando la librería Jena, leemos la ontología y calculamos todas sus propiedades. El programa recibe como

parámetro un fichero con una ontología OWL, y muestra por pantalla los valores de las propiedades calculadas.

- **QueryGeneration**: Este programa también utiliza la librería Jena para procesar las ontologías. Con la información anotada en los conceptos de la ontología, se generan consultas SQL sobre el atributo de texto indicado en los parámetros. Los parámetros del programa están incluidos en un fichero de configuración, que contiene los detalles de la conexión a la BD, la tabla y atributo sobre el que realizar la consulta, y el concepto por el que queremos consultar. Si no se especifica un concepto de consulta, se crea una consulta para el concepto raíz. La forma de crear la consulta es sencilla, tomamos el concepto por el que se va a consultar y lo localizamos en la ontología. Tomamos el concepto y todos sus subconceptos, y extraemos de ellos las anotaciones correspondientes. Creamos una consulta SQL sobre la tabla y atributo indicados en los parámetros, estableciendo como criterio de selección que aparezca alguno de los términos extraídos de las anotaciones. Para determinar la cobertura, se realiza la división del número de tuplas obtenidas, entre el número de tuplas totales. Esto nos da un valor de cobertura normalizado en el intervalo $[0, 1]$.

También hemos realizado algunos programas adicionales para la ejecución por lotes de los distintos algoritmos y su evaluación. Dado que estos no tienen más utilidad que la de generar los datos para los experimentos, no consideramos que sea necesario describirlos aquí.

Finalmente, para la realización de los tests estadísticos, hemos usado la herramienta descrita en [García and Herrera, 2008].

B.8. WOntologyGenerator API

La mayor parte de las herramientas desarrolladas están incluidas en la API *WOntologyGenerator*, que hemos desarrollado. Esta API está dividida en diversos paquetes que proporcionan funcionalidades para la generación de ontologías utilizando WordNet/Wikipedia. Actualmente abarca la funcionalidad relacionada con las etapas de preprocesamiento semántico y extensión semántica.

Las herramientas correspondientes a las etapas de preprocesamiento sintáctico y generación de formas intermedias basadas en Conjuntos-AP, existían previamente y aún no han sido integradas dentro de la API, si bien ésta podría estructurarse de forma más modular para adaptarse a cada uno de las etapas de la metodología.

Vamos a realizar un repaso rápido a cada uno de los paquetes que componen la API:

- **es.ugr.citic.wog.apsets** - Este paquete contiene las estructuras necesarias para el manejo de los Conjuntos-AP. Contiene clases para la gestión de las Estructuras-AP en memoria y sus itemsets frecuentes.

- **es.ugr.citic.wog.configuration** - Este paquete contiene todas las utilidades necesarias para gestionar los parámetros de configuración leídos de fichero.

- **es.ugr.citic.wog.database** - Se encarga de la gestión de las conexiones con la base de datos.
- **es.ugr.citic.wog.exceptions** - Gestión de las excepciones del programa.
- **es.ugr.citic.wog.files** - Gestión de ficheros de salida.
- **es.ugr.citic.wog.ir.wikipedia** - Extensión de la librería de recuperación de información para tratar con el contenido textual de las categorías de Wikipedia.
- **es.ugr.citic.wog.measures** - Implementación de las medidas utilizadas. Están agrupadas por tipo de medidas, tenemos las interfaces:
  - *SetMetrics*, para métricas basadas en evaluación de conjuntos, tales como Jaccard, Sorensen-Dice, Mountford, Overlap.
  - *TopologicalMetrics*, basados en medidas de distancia sobre ontologías tales como Rada, Leacock-Chodorow y Wu-Palmer.
  - *ICMetrics* para medidas basadas en contenido de información como Resnik, Seco, Lin, Jiang-Conrath.
  - *IRMetrics* para medidas basadas en recuperación de información, tales como Coseno.
- **es.ugr.citic.wog.ontology** - Permite la gestión de ontologías, lectura, creación, edición, escritura, etc. Se puede utilizar para generar ontologías a partir de formas de representación intermedias, sin el uso de herramientas externas. Actualmente la única representación intermedia soportada es la de los Conjuntos-AP.
- **es.ugr.citic.wog.ontology.query** - Se usa para generar consultas SQL a partir de la ontología.
- **es.ugr.citic.wog.ontology.statistics** - Permite el cálculo de datos estadísticos y propiedades estructurales sobre una ontología. Con este paquete podemos obtener datos tales como número de nodos de una ontología, profundidad, anchura, complejidad, dispersión, etc.
- **es.ugr.citic.wog.ontology.wikipedia** - En este paquete se encuentran todas las utilidades necesarias para generar ontologías OWL a partir de los Conjuntos-AP utilizando Wikipedia. Todos los métodos de generación de ontologías empleando Wikipedia están reunidos aquí.
- **es.ugr.citic.wog.ontology.wordnet** - El paquete contiene todas las utilidades para procesar ontologías OWL usando WordNet. Por un lado, tenemos la generación de ontologías usando WordNet y por otro, la extensión de los conceptos de una ontología con sinónimos obtenidos de WordNet.
- **es.ugr.citic.wog.postagger** - Aquí creamos un recubrimiento, para acceder de forma sencilla desde nuestro código a la API del etiquetador de categoría gramatical de Stanford.

- **es.ugr.citic.wog.synsetsubstitution** - Las clases para la realización del preprocesamiento semántico se encuentran en este paquete. Concretamente, para las tareas de agrupamiento de sinónimos.
- **es.ugr.citic.wog.synsetsubstitution.wordfrequencies** - Contiene las estructuras y procedimientos necesarios para la gestión de las frecuencias de los términos desambiguados.
- **es.ugr.citic.wog.tree** - Aquí podemos encontrar la definición de un TDA para un árbol genérico. Lo usaremos para mantener las categorías seleccionadas en memoria, antes de proceder a convertirlas en una ontología.
- **es.ugr.citic.wog.wikipedia** - Recubrimiento sobre la API JWPL para el acceso a Wikipedia, para poder acceder de forma sencilla desde nuestro código.
- **es.ugr.citic.wog.wordnet** - Recubrimiento sobre la API JWNL de acceso a WordNet, para poder acceder de forma sencilla desde nuestro código.
- **es.ugr.citic.wog.wordnet.disambiguation** - Implementación de los algoritmos de desambiguación.

De forma adicional se han desarrollado diferentes programas que se incluyen con la API como son:

- **SemanticPreprocessor** - Programa para la realización de la etapa de preprocesamiento semántico de la ontología.
- **WordNetOntologyCreator** - Programa para la generación de ontologías a partir de Conjuntos-AP usando WordNet.
- **WikipediaOntologyCreator** - Programa para la generación de ontologías a partir de Conjuntos-AP usando Wikipedia.
- **Evaluation** - Programa de cálculo de medidas estructurales sobre ontologías.
- **QueryGeneration** - Generación de consultas a partir de las ontologías extendidas generadas.

Mediante el uso de esta API es posible integrar los procedimientos que hemos desarrollado en otras plataformas software.

B.9. Conclusiones

En este capítulo hemos hecho un breve repaso al software desarrollado durante la elaboración de esta Tesis.

Hemos presentado la herramienta para transformar una ontología OWL en una secuencia de comandos SQL para crear las estructuras necesarias en la base de datos, y también

hemos presentado el software utilizado durante cada una de las etapas de la metodología de extracción de semántica a partir de texto.

Hemos presentado una herramienta desarrollada para realizar el preprocesamiento sintáctico de los datos, lo cual facilita un posterior proceso de minería de datos o de texto. El origen de los datos es una base de datos relacional y los conjuntos de datos generados pueden ser almacenados como tablas o escritos a un fichero. La herramienta permite aplicar diferentes filtros para procesado de texto. La automatización del preprocesamiento de los datos, facilita la generación de diferentes conjuntos de datos. Debido a las necesidades específicas del procesamiento que hemos debido realizar, la mejor opción era la implementación de esta herramienta, genérica y extensible.

Para la etapa de preprocesamiento semántico hemos desarrollado un programa, que usando librerías para el acceso a WordNet, y un etiquetador de categoría gramatical, permite detectar y agrupar términos en conjuntos de sinónimos.

Adaptando y ampliando una herramienta existente, hemos realizado el cálculo de los Conjuntos-AP para atributos textuales en bases de datos Oracle^{TM}.

También hemos desarrollado software para la creación de ontologías OWL a partir de Estructuras-AP usando Wikipedia y WordNet. Para poder realizar la evaluación de los resultados obtenidos, hemos creados unos programas que nos proporcionan datos sobre la estructura de la ontología y la cobertura de ésta sobre el conjunto de datos inicial.

Finalmente todos los nuevos desarrollos realizados se han integrado en una API.

Como trabajo futuro queda pendiente la integración de todos los elementos en un interfaz gráfico, un plug-in de Protégé, por ejemplo, para poder realizar todo el procesamiento de forma sencilla.

Bibliografía

[OWL, 2004] (2004). OWL Web Ontology Language. http://www.w3.org/TR/owl-ref/.

[OMD, 2005] (2005). Ontology definition metamodel. Third Revised Submission to OMG/ RFP ad/2003-03-40.

[Pro, 2008] (2008). The Protégé ontology editor and knowledge acquisition system. http://protege.stanford.edu/.

[Agrawal and R.Srikant, 1994] Agrawal, R. and R.Srikant (1994). Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, pages 487–499.

[Al-Jadir et al., 2010] Al-Jadir, L., Parent, C., and Spaccapietra, S. (2010). Reasoning with large ontologies stored in relational databases: The OntoMinD approach. *Data & Knowledge Engineering*.

[Alexaki et al., 2001] Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D., and Tolle, K. (2001). The ICS-FORTH RDFSuite: Managing voluminous RDF description bases. In *2nd International Workshop on the Semantic Web (SemWeb'01)*, pages 1–13.

[An et al., 2005] An, Y., Borgida, A., and Mylopoulos, J. (2005). Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, pages 1152–1169.

[An et al., 2006] An, Y., Mylopoulos, J., and Borgida, A. (2006). Building semantic mappings from databases to ontologies. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, pages 1557–1560.

[Andreasen and Bulskov, 2009] Andreasen, T. and Bulskov, H. (2009). Conceptual querying through ontologies. *Fuzzy Sets and Systems*, 160(15):2159 – 2172. Special Issue: The Application of Fuzzy Logic and Soft Computing in Information Management.

[Astrova et al., 2007] Astrova, I., Korda, N., and Kalja, A. (2007). Storing owl ontologies in sql relational databases. *International Journal of Electrical, Computer, and Systems Engineering*, 1(4):242–247.

[Auer et al., 2008] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2008). Dbpedia: A nucleus for a web of open data. *The Semantic Web*, pages 722–735.

[Baader and Werner, 2003] Baader, F. and Werner, N. (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*, chapter Basic Description Logics, pages 47–100. Cambridge University Press, New York, NY, USA.

[Bagui, 2009] Bagui, S. (2009). Mapping OWL to the Entity Relationship and Extended Entity Relationship models. *International Journal of Knowledge and Web Intelligence*, 1(1):125–149.

[Bagui and Earp, 2003] Bagui, S. and Earp, R. (2003). *Database design using entity-relationship diagrams*. Auerbach Publications.

[Banerjee and Pedersen, 2002] Banerjee, S. and Pedersen, T. (2002). An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet. In *Proceedings of the Conference on Computational Linguistics and Intelligent Text Processing (CICLING)*, pages 136–145. Springer.

[Banerjee and Pedersen, 2003] Banerjee, S. and Pedersen, T. (2003). Extended gloss overlaps as a measure of semantic relatedness. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 805–810.

[Barranco et al., 2009a] Barranco, C., Campaña, J., and Medina, J. (2009a). A Real Estate Management System Based on Soft Computing. *Applications of Soft Computing*, pages 31–40.

[Barranco et al., 2008a] Barranco, C. D., Campaña, J. R., and Medina, J. M. (2008a). A b+-tree based indexing technique for fuzzy numerical data. *Fuzzy Sets and Systems*, 159(12):1431–1449. Advances in Intelligent Databases and Information Systems.

[Barranco et al., 2009b] Barranco, C. D., Campaña, J. R., and Medina, J. M. (2009b). Indexing fuzzy numerical data with a B+ tree for fast retrieval using necessity-measured flexible conditions. *International Journal Uncertainty, Fuzziness and Knowledge Based Systems*, 17(supp01):1–23.

[Barranco et al., 2005a] Barranco, C. D., Campaña, J. R., Cubero, J. C., and Medina, J. M. (2005a). A fuzzy object relational approach to flexible real estate trade. *WSEAS Transactions on Information Science and Applications*, 2(2):155–160.

[Barranco et al., 2005b] Barranco, C. D., Campaña, J. R., and Medina, J. M. (2005b). Improving query expressiveness in product search interfaces using fuzzy logic. *WSEAS Transactions on Business and Economics*, 2(2):80–87.

[Barranco et al., 2008b] Barranco, C. D., Campaña, J. R., and Medina, J. M. (2008b). *Handbook of Research on Fuzzy Information Processing in Databases*, chapter Towards a Fuzzy Object-Relational Database Model, pages 431–461. Hershey, PA, USA, first edition.

[Barranco et al., 2004] Barranco, C. D., Campaña, J. R., Medina, J. M., and Pons, O. (2004). ImmoSoftWeb: a web based fuzzy application for real estate management. *Lectures Notes in Artificial Intelligence(LNAI)*, 3034:196–206.

[Barranco et al., 2007] Barranco, C. D., Campaña, J. R., Medina, J. M., and Pons, O. (23-26 July 2007). On storing ontologies including fuzzy datatypes in relational databases. *Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International*, pages 1–6.

[Barrasa et al., 2003] Barrasa, J., Corcho, O., and Gómez-Pérez, A. (2003). Fund Finder: A case study of database-to-ontology mapping. In *Proc. ISWC Semantic integration workshop, Sanibel Island, Florida.*

[Barrasa et al., 2004] Barrasa, J., Corcho, O., and Gómez-Pérez, A. (2004). R2O, an extensible and semantically based database-to-ontology mapping language. In *Proceedings of the Second Workshop on Semantic Web and Databases (SWDB2004).*

[Batini et al., 1992] Batini, C., Ceri, S., and Navathe, S. B. (1992). *Conceptual Database Design: An Entity-Relationship Approach.* Benjamin/Cummings.

[Beckett, 2002] Beckett, D. (2002). The design and implementation of the Redland RDF application framework. *Computer Networks*, 39(5):577–588.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5):28–37.

[Bizer, 2003] Bizer, C. (2003). D2R MAP-A database to RDF mapping language. *WWW (Posters).*

[Bizer and Cyganiak, 2006] Bizer, C. and Cyganiak, R. (2006). D2R Server - Publishing Relational Databases on the Semantic Web. In *5th International Semantic Web Conference.*

[Bizer and Seaborne, 2004] Bizer, C. and Seaborne, A. (2004). D2RQ-Treating Non-RDF Databases as Virtual RDF Graphs. In *Proceedings of the 3rd International Semantic Web Conference (ISWC2004).*

[Borgida et al., 2003] Borgida, A., Lenzerini, M., and Rosati, R. (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*, chapter Description Logics for Databases, pages 462–484. Cambridge University Press, New York, NY, USA.

[Bozsak et al., 2002] Bozsak, E., Ehrig, M., Handschuh, S., Hotho, A., Maedche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S., Stojanovic, L., et al. (2002). KAONTowards a large scale semantic web. *E-Commerce and Web Technologies*, pages 231–248.

[Brank et al., 2005] Brank, J., Grobelnik, M., and Mladenic, D. (2005). A survey of ontology evaluation techniques. In *Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005)*, pages 166–170.

[Brants, 2000] Brants, T. (2000). TnT: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231.

[Broekstra et al., 2002] Broekstra, J., Kampman, A., and Van Harmelen, F. (2002). Sesame: A generic architecture for storing and querying rdf and rdf schema. In *Proceedings of the First International Semantic Web Conference (ISWC 2002)*, volume 2342, pages 54–68.

[Broekstra et al., 2003] Broekstra, J., Kampman, A., and Van Harmelen, F. (2003). *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, chapter Sesame: An architecture for storing and querying RDF data and schema information, pages 197–222.

[Buckles and Petry, 1982a] Buckles, B. P. and Petry, F. E. (1982a). Fuzzy databases and their applications. *Fuzzy Information and Decision Processes*, 2:361–371.

[Buckles and Petry, 1982b] Buckles, B. P. and Petry, F. E. (1982b). A fuzzy representation of data for relational databases. *Fuzzy Sets and Systems*, 7:213–226.

[Calvanese et al., 1998] Calvanese, D., Lenzerini, M., and Nardi, D. (1998). *Description logics for conceptual data modeling*, chapter 8, pages 229–264. Kluwer Academic Publisher.

[Campaña et al., 2009] Campaña, J., Martín-Bautista, M., Medina, J., and Vila, M. (2009). Semantic Enrichment of Database Textual Attributes. *Flexible Query Answering Systems, FQAS 2009*, pages 488–499.

[Chen, 1976] Chen, P. (1976). The Entity-Relationship Model – Toward a Unified View of Data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36.

[Cimiano, 2006] Cimiano, P. (2006). *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*, chapter Ontology Learning from Text, pages 19–34. Springer.

[Cimiano and Staab, 2005] Cimiano, P. and Staab, S. (2005). Learning concept hierarchies from text with a guided hierarchical clustering algorithm. In *ICML workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods*.

[Codd, 1970] Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.

[Cubero et al., 2004] Cubero, J. C., Marín, N., Medina, J. M., Pons, O., and Vila, M. A. (2004). Fuzzy object management in an object-relational framework. In *X Intl. Conf. of information processing and management of uncertainty in knowledge-based systems*, pages 1767–1774.

[Cullot et al., 2007] Cullot, N., Ghawi, R., and Yétongnon, K. (2007). DB2OWL: A Tool for Automatic Database-to-Ontology Mapping. In *Proceedings of 15th Italian Symposium on Advanced Database Systems (SEBD 2007)*, pages 491–494.

[Cunningham, 2002] Cunningham, H. (2002). GATE, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254.

[Das et al., 2004] Das, S., Chong, E. I., Eadon, G., and Srinivasan, J. (2004). Supporting ontology-based semantic matching in RDBMS. In *Proceedings of the* 30^{th} *VLDB Conference, Toronto, Canada*, pages 1054–1065.

[Date, 1990] Date, C. (1990). *An Introduction to Database Systems*. Addison-Wesley, Reading, MA.

[Dehainsala et al., 2007] Dehainsala, H., Pierra, G., and Bellatreche, L. (2007). Ontodb: An ontology-based database for data intensive applications. In Kotagiri, R., Krishna, P., Mohania, M., and Nantajeewarawat, E., editors, *Advances in Databases: Concepts, Systems and Applications*, volume 4443 of *Lecture Notes in Computer Science*, pages 497–508. Springer Berlin / Heidelberg.

[Dice, 1945] Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302.

[Erling and Mikhailov, 2009] Erling, O. and Mikhailov, I. (2009). Rdf support in the virtuoso dbms. In Pellegrini, T., Auer, S., Tochtermann, K., and Schaffert, S., editors, *Networked Knowledge - Networked Media*, volume 221 of *Studies in Computational Intelligence*, pages 7–24. Springer Berlin / Heidelberg.

[Fellbaum, 1998] Fellbaum, C. (1998). *WordNet: an electronic lexical database*. MIT Press, Cambridge, MA.

[Finkelstein et al., 2002] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2002). Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

[Friedman, 1937] Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701.

[Friedman, 1940] Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92.

[Fukami et al., 1979] Fukami, S., Umano, M., Muzimoto, M., and Tanaka, H. (1979). Fuzzy database retrieval and manipulation language. *IEICE Technical Reports*, 78(233):65–72.

[Gabrilovich and Markovitch, 2007] Gabrilovich, E. and Markovitch, S. (2007). Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1606–1611.

[Gangemi et al., 2006] Gangemi, A., Catenacci, C., Ciaramita, M., and Lehmann, J. (2006). Modelling ontology evaluation and validation. *The Semantic Web: Research and Applications*, pages 140–154.

[García and Herrera, 2008] García, S. and Herrera, F. (2008). An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons. *Journal of Machine Learning Research*, 9(2677–2694).

[Giménez and Márquez, 2004] Giménez, J. and Márquez, L. (2004). Svmtool: A general pos tagger generator based on support vector machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 43–46.

[Gruber, 1995] Gruber, T. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human Computer Studies*, 43(5):907–928.

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220.

[Guarino and Giaretta, 1995] Guarino, N. and Giaretta, P. (1995). Ontologies and knowledge bases. *Towards Very Large Knowledge Bases*.

[Guha and McCool, 2003] Guha, R. and McCool, R. (2003). TAP: a Semantic Web platform. *Computer Networks*, 42(5):557–577.

[Guha, 2000] Guha, R. V. (2000). rdfDB: An RDF database. http://guha.com/rdfdb.

[Handschuh and Staab, 2002] Handschuh, S. and Staab, S. (2002). Authoring and annotation of web pages in CREAM. In *Proceedings of the 11th international conference on World Wide Web*, pages 462–473.

[Handschuh et al., 2002] Handschuh, S., Staab, S., and Ciravegna, F. (2002). S-CREAMsemi-automatic creation of metadata. In Gómez-Pérez, A. and Benjamins, V., editors, *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, volume 2473 of *Lecture Notes in Computer Science*, pages 165–184. Springer Berlin / Heidelberg.

[Handschuh et al., 2001] Handschuh, S., Staab, S., and Maedche, A. (2001). CREAM: creating relational metadata with a component-based, ontology-driven annotation framework. In *Proceedings of the 1st international conference on Knowledge capture*, pages 76–83.

[Harris and Gibbins, 2003] Harris, S. and Gibbins, N. (2003). 3store: Efficient bulk RDF storage. In *Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PSSS03)*, pages 1–20. Citeseer.

[Harris, 1968] Harris, Z. S. (1968). *Mathematical Structures of Language*. Wiley.

[Hearst, 1992] Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, volume 2, pages 539–545. Association for Computational Linguistics Morristown, NJ, USA.

[Hirst and St.Onge, 1998] Hirst, G. and St.Onge, D. (1998). *Lexical Chains as representation of context for the detection and correction malapropisms*. The MIT Press.

[Hitzler et al., 2009] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S., editors (27 October 2009). *OWL2 Web Ontology Language: Primer*. W3C Recommendation. http://www.w3.org/TR/owl2-primer/.

[Hofmann, 1999] Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 50–57. ACM Press New York, NY, USA.

[Hommel, 1988] Hommel, G. (1988). A stagewise rejective multiple test procedure based on a modified Bonferroni test. *Biometrika*, 75(2):383.

[Horrocks et al., 2004] Horrocks, I., Li, L., Turi, D., and Bechhofer, S. (2004). The instance store: DL reasoning with large numbers of individuals. In *2004 International Workshop on Description Logics*, pages 31–40.

[Jaccard, 1901] Jaccard, P. (1901). Etude comparative de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Société vaudoise des Sciences Naturelles*, (37):547–579.

[Jean et al., 2006] Jean, S., Pierra, G., and Ait-Ameur (2006). Domain ontologies: a database-oriented analysis. In *Web Information Systems and Technologies (WEBIST 2006)*, pages 238–254.

[Jiang and Conrath, 1997] Jiang, J. and Conrath, D. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the 10th international conference on research in computational linguistics, Taipei, Taiwan*, pages 19–33.

[Kalyanpur et al., 2006] Kalyanpur, A., Parsia, B., Sirin, E., Grau, B. C., and Hendler, J. (2006). Swoop: A web ontology editing browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(2):144–153.

[Konstantinou et al., 2006] Konstantinou, N., Spanos, D., Chalas, M., Solidakis, E., and Mitrou, N. (2006). VisAVis: An approach to an intermediate layer between ontologies and relational database contents. In *International Workshop on Web Information Systems Modeling (WISM2006) Luxembourg*.

[Laclavík, 2006] Laclavík, M. (2006). RDB2Onto: Relational Database Data to Ontology Individual Mapping. In *Tools for Acquisition, Organisation and Presenting of Information and Knowledge*, pages 86–89.

[Landauer and Dumais, 1997] Landauer, T. K. and Dumais, S. T. (1997). A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, 104:211–240.

[Leacock and Chodorow, 1998] Leacock, C. and Chodorow, M. (1998). *Combining Local Context and WordNet Similarity for Word Sense Identification*, chapter 11, pages 265–283. The MIT Press.

[Lesk, 1986] Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the ACM-SIGDOC Conference, Toronto, Canada*, pages 24–26. ACM New York, NY, USA.

[Li, 2004] Li, L. (2004). Reasoning with large numbers of individuals moves on: extending the instance store.

[Lin, 1998] Lin, D. (1998). An information-theoretic definition of similarity. In Shavlik, J. W., editor, *Proc. 15th International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann.

[Madche and Staab, 2001] Madche, A. and Staab, S. (2001). Ontology learning for the Semantic Web. *IEE Intelligent Systems*, 16(2):72–79.

[Maedche et al., 2002] Maedche, A., Motik, B., Silva, N., and Volz, R. (2002). MAFRAa mapping framework for distributed ontologies. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 69–75.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

[Marcus et al., 1993] Marcus, M., Marcinkiewicz, M., and Santorini, B. (1993). Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

[Marín et al., 2006] Marín, N., Martín-Bautista, M. J., Prados, M., and Vila, M. A. (2006). Enhancing Short Text Retrieval in Databases. In *Proceedings of FQAS 2006, Milan, Italy. Lecture Notes in Artificial Intelligence.*, volume 4027, pages 613–624. Springer.

[Martín-Bautista et al., 2008] Martín-Bautista, M. J., Martínez-Folgoso, S., and Vila, M. (2008). A New Semantic Representation for Short Texts. In *Data Warehousing and Knowledge Discovery: 10th International Conference, Dawak 2008 Turin, Italy, September 1-5, 2008*, pages 347–356. Springer.

[Martín-Bautista et al., 2006] Martín-Bautista, M. J., Prados, M., Vila, M. A., and Martinez-Folgoso, S. (2006). A knowledge representation for short texts based on frequent itemsets. In *Proceedings of IPMU 2006, Paris, France.*, pages 1065–1070.

[Martínez-Folgoso, 2008] Martínez-Folgoso, S. (2008). Tratamiento semántico de atributos textuales en un modelo relacional orientado a objetos: Implementación en software libre. Ph.D. Thesis.

[Medina et al., 2002] Medina, J. M., Galindo, J., Berzal, F., and Serrano, J. M. (2002). Using object relational features to build a fuzzy database server. In *VIII Intl. Conf.*

of information processing and management of uncertainty in knowledge-based systems (IPMU 2002), pages 307–314.

[Medina et al., 1994] Medina, J. M., Pons, O., and Vila, M. A. (1994). GEFRED: A generalized model of fuzzy relational databases. *Information Sciences*, 76(1-2):87–109.

[Mehler, 2009] Mehler, A. (2009). A quantitative graph model of social ontologies by example of Wikipedia. *Genres on the Web: Computational Models and Empirical Studies*, pages 291–352.

[Miller and Charles, 1991] Miller, G. and Charles, W. (1991). Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

[Milne and Witten, 2008] Milne, D. and Witten, I. (2008). An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, pages 25–30.

[Mountford, 1962] Mountford, M. D. (1962). *An index of similarity and its application to classification problems*, pages 43–50. P.W.Murphy editors.

[Navathe and Elmasri, 2007] Navathe, S. and Elmasri, R. (2007). *Fundamentals of Database Systems*. Addison Wesley.

[Pan and Heflin, 2003] Pan, Z. and Heflin, J. (2003). DLDB: Extending relational databases to support semantic web queries. In *Workshop on Practical and Scaleable Semantic Web Systms, ISWC*, pages 109–113.

[Pan et al., 2008] Pan, Z., Zhang, X., and Heflin, J. (2008). Dldb2: A scalable multi-perspective semantic web repository. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, volume 1, pages 489–495. IEEE.

[Patwardhan et al., 2003] Patwardhan, S., Banerjee, S., and Pedersen, T. (2003). Using measures of semantic relatedness for word sense disambiguation. In Gelbukh, A., editor, *Computational Linguistics and Intelligent Text Processing (CICLing 2003)*, volume 2588 of *Lecture Notes in Computer Science*, pages 241–257. Springer Berlin / Heidelberg.

[Porter, 1980] Porter, M. (1980). An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137.

[Prade and Testemale, 1984] Prade, H. and Testemale, C. (1984). Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. *Information Sciences*, 34:115–143.

[Rada et al., 1989] Rada, R., Mili, H., Bicknell, E., and Blettner, M. (1989). Development and application of a metric on semantic nets. *IEEE Transactions on Systems Management and Cybernetics*, 19(1):17–30.

[Resnik, 1995] Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *In Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453.

[Resnik, 1999] Resnik, P. (1999). Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of artificial intelligence research*, 11(95):130.

[Roldán García et al., 2005] Roldán García, M., Navas Delgado, I., and Aldana Montes, J. (2005). A design methodology for semantic web database-based systems. In *ICITA (1)*, pages 233–237.

[Rubenstein and Goodenough, 1965] Rubenstein, H. and Goodenough, J. (1965). Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

[Salton and Buckley, 1988] Salton, G. and Buckley, C. (1988). Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5):515–523.

[Salton and McGill, 1983] Salton, G. and McGill, M. (1983). *Introduction to modern information retrieval*. McGraw-Hill New York.

[Schmid, 1999] Schmid, H. (1999). Improvements in part-of-speech tagging with an application to German. *Natural language processing using very large corpora*, 11:13–26.

[Seco et al., 2004] Seco, N., Veale, T., and Hayes, J. (2004). An intrinsic information content metric for semantic similarity in WordNet. In *Proceedings of ECAI-04*, volume 16, pages 1089–1090.

[Sorensen, 1948] Sorensen, T. (1948). A method of establishing groups of equal amplitude in plant sociology based on similarity ofspecies content and its application to analyses of the vegetation on danish commons. *Dan. Vidensk. Selsk. Biol. Skr.*, (5):1–34.

[Stoffel et al., 1997] Stoffel, K., Taylor, M., and Hendler, J. (1997). Efficient management of very large ontologies. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 442–447. JOHN WILEY & SONS LTD.

[Stojanovic et al., 2002a] Stojanovic, L., Stojanovic, N., and Volz, R. (2002a). Migrating data-intensive web sites into the semantic web. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 1100–1107. ACM.

[Stojanovic et al., 2002b] Stojanovic, N., Stojanovic, L., and Volz, R. (2002b). A reverse engineering approach for migrating data-intensive web sites to the Semantic Web. In *Intelligent Information Processing, IFIP 17th World Computer CongressTC12 Stream on Intelligent Information Processing, Montreal*. Kluwer Academic Publishers.

[Strube and Ponzetto, 2006] Strube, M. and Ponzetto, S. (2006). WikiRelate! Computing semantic relatedness using Wikipedia. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, pages 1419–1424.

[Sugumaran and Storey, 2006] Sugumaran, V. and Storey, V. C. (2006). The role of domain ontologies in database design: An ontology management and conceptual modeling environment. *ACM Trans. Database Syst.*, 31(3):1064–1094.

[Teorey, 1989] Teorey, T. J. (1989). Distributed database design: A practical approach and example. *SIGMOD Record*, 18(4):23–39.

[Thalheim, 1993] Thalheim, B. (1993). Foundations of entity - relationship modeling. *Ann. Math. Artif. Intell.*, 7(1-4):197–256.

[Theoharis et al., 2005] Theoharis, Y., Christophides, V., and Karvounarakis, G. (2005). Benchmarking database representations of rdf/s stores. In Gil, Y., Motta, E., Benjamins, V., and Musen, M., editors, *International Semantic Web Conference (ISWC 2005)*, volume 3729 of *Lecture Notes in Computer Science*, pages 685–701. Springer Berlin / Heidelberg.

[Torsten Zesch, 2007] Torsten Zesch, I. G. (2007). Analysis of the wikipedia category graph for nlp applications. pages 1–8.

[Toutanova et al., 2003] Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259.

[Toutanova and Manning, 2000] Toutanova, K. and Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pages 63–70.

[Trinh et al., 2006] Trinh, Q., Barker, K., and Alhajj, R. (2006). RDB2ONT: A tool for generating owl ontologies from relational database systems. In *Advanced International Conference on Telecommunications (AICT 2006)*.

[Trinkunas and Vasilecas, 2007] Trinkunas, J. and Vasilecas, O. (2007). A graph oriented model for ontology transformation into conceptual data model. *INFORMATION TECHNOLOGY AND CONTROL*, 36(1A):126–132.

[Turney, 2001] Turney, P. D. (2001). Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning (ECML)*, pages 491–502. Springer-Verlag.

[Umano, 1982] Umano, M. (1982). Freedom-O: A fuzzy database system. *Fuzzy Information and Decision Processes*.

[Vasilescu et al., 2004] Vasilescu, F., Langlais, P., and Lapalme, G. (2004). Evaluating variants of the Lesk approach for disambiguating words. In *Proceedings of the Conference of Language Resources and Evaluations (LREC 2004)*, pages 633–636.

[Volz et al., 2004] Volz, R., Handschuh, S., Staab, S., Stojanovic, L., and Stojanovic, N. (2004). Unveiling the Hidden Bride: Deep Annotation for Mapping and Migrating Legacy Data to the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(2):187–206.

[Vrandecic, 2009] Vrandecic, D. (2009). *Handbook on Ontologies, Second Edition*, chapter Ontology Evaluation, pages 315–336.

[Vysniauskas and Nemuraite, 2006] Vysniauskas, E. and Nemuraite, L. (2006). Transforming ontology representation from OWL to relational database. *Information Technology And Control*, 35A(3):333–343.

[W3C OWL Working Group, 2009] W3C OWL Working Group (27 October 2009). *OWL2 Web Ontology Language: Document Overview.* W3C Recommendation. http://www.w3.org/TR/owl2-overview/.

[Wikipedia, 2010] Wikipedia (2010). Wikipedia, the free encyclopedia. http://wikipedia.org/. [Online; accessed 22-October-2010].

[Wood, 2005] Wood, D. (2005). Scaling the kowari metastore. In Dean, M., Guo, Y., Jun, W., Kaschek, R., Krishnaswamy, S., Pan, Z., and Sheng, Q., editors, *Web Information Systems Engineering WISE 2005 Workshops*, volume 3807 of *Lecture Notes in Computer Science*, pages 193–198. Springer Berlin / Heidelberg.

[Wood et al., 2005] Wood, D., Gearon, P., and Adams, T. (2005). Kowari: A platform for semantic web storage and analysis. In *Proceedings of XTech 2005 Conference.*

[Wu and Palmer, 1994] Wu, Z. and Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138, Morristown, NJ, USA. Association for Computational Linguistics.

[Yu et al., 2009] Yu, J., Thom, J., and Tam, A. (2009). Requirements-oriented methodology for evaluating ontologies. *Information Systems*, 34(8):766–791.

[Zadeh, 1965] Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3):338–353.

[Zemankova-Leech and Kandel, 1984] Zemankova-Leech, M. and Kandel, A. (1984). *Fuzzy Relational Databases – A Key to Expert Systems.* Köln, Germany, TÜV Rheinland.

[Zemankova-Leech and Kandel, 1985] Zemankova-Leech, M. and Kandel, A. (1985). Implementing imprecision in information systems. *Information Sciences*, 37:107–141.

[Zesch and Gurevych, 2010] Zesch, T. and Gurevych, I. (2010). Wisdom of crowds versus wisdom of linguists–measuring the semantic relatedness of words. *Natural Language Engineering*, 16(01):25–59.

[Zhou et al., 2006] Zhou, J., Ma, L., Liu, Q., Zhang, L., Yu, Y., and Pan, Y. (2006). Minerva: A scalable OWL ontology storage and inference system. volume 4185 of *Lecture Notes in Computer Science*, pages 429–443. Springer Berlin / Heidelberg.

[Zobel and Moffat, 1998] Zobel, J. and Moffat, A. (1998). Exploring the similarity space. In *ACM SIGIR Forum*, volume 32, pages 18–34.

UNIVERSITY OF GRANADA

SCHOOL OF COMPUTER AND TELECOMMUNICATIONS ENGINEERING

Department of Computer Science
and Artificial Intelligence

Semantic Representation and Management of Imprecise Information in Databases

Ph.D. Thesis Summary

Jesús R. Campaña Gómez

Advisors: Juan Miguel Medina Rodríguez y María Amparo Vila Miranda

Granada, May 2011

Preface

This document contains a summarized version of the Ph.D. Thesis **"Representación y tratamiento semántico de Información Imprecisa en Bases de Datos" (Semantic Representation and Management of Imprecise Information in Databases)**. This summary is one of the requirements to apply for the European Doctor mention of the Doctoral Thesis.

This document summarizes the main document of the Ph.D. Thesis which is written in Spanish. The thesis studies different approaches to deal with information semantics in a relational database environment.

The original Ph.D. Thesis contents are structured as follows:

An introduction and a motivation of our work is presented in Chapter 1. Chapter 2, presents a comprehensive review of the topics studied. We present a review on different techniques dealing with semantics in databases. This review is structured in three parts, ontology storage in databases, semantic extraction from text attributes and semantic similarity/relatedness measures. Regarding ontology storage, we discuss ontologies as conceptual design tools for databases, ontology storage in relational databases, ontology extraction from relational databases and ontology storage systems. The semantic extraction from texts review is devoted to discuss different approaches for ontology learning from texts, ontology learning using WordNet and to describe a intermediate representation form for texts. This text representation called AP-Sets is used in our proposal and thus is described Section 3.2.2 of this document. To conclude the chapter, we review different approaches to determine semantic relatedness in taxonomies, and the use of ontologies to extend queries.

The following chapters contain the bulk of our proposal, and thus their content is summarized as part of this document. Chapter 3 deals with ontology storage in relational databases, the creation of schemas using ontologies and how to store ontology definitions and instances together. This chapter is summarized in this document as Chapter 2. Chapter 4 presents a methodology for semantic extension of non-structured texts along with an implementation of the methodology using WordNet. Chapter 5 presents an implementation of the methodology using Wikipedia and a study of different methods to extend texts semantics using Wikipedia. Both of these chapters are summarized in this document as Chapter 3.

Chapter 6 presents a experimental evaluation of the methods used to extend semantics of texts using Wikipedia. Finally, Chapter 7 gives some conclusions and highlight future work.

The main chapters are complemented by two Appendixes. Appendix A contains a brief introduction to some of the topics discussed in our work, such as imprecise databases, ontologies and WordNet. Appendix B gives a brief description of the software developed in the course of our study, and of some other software used by us and developed within our research group.

Contents

Chapter 1

Introduction

Nowadays information has become an asset for people and companies across the world. Massive use of Internet has brought the emergence of new data sources and new uses for information. Management of current volumes of information is a very complex task. Traditionally, database management systems have been used for the management of huge volumes of data. Although data management paradigm is evolving; it is not as important to have huge quantities of data, as the management of the knowledge coded on data.

Recent years have seen the emergence of new ways to manage data. Data storage and recovery is not enough; data must be processed and understood in order to make the best use of it. Automatic processing and reasoning over data is a hot topic today. Traditionally, the most used knowledge representation has been that of ontologies. The use of ontologies as a knowledge representation formalism is taking center stage again, due to the growing importance of Semantic Web technologies.

On this scenario we have databases, which have been successfully used for data management on the classical web, and ontologies, which are of key importance to enable the Semantic Web. The idea of combining both technologies comes naturally. Evolution from a web of data to a web of knowledge needs the use of representation formalisms that ontologies provide. But, after all, ontologies are data that needs to be managed, stored, and processed.

In this work, we propose to study different approaches to deal with information semantics in a relational database environment. Our objective is to provide users with tools to deal with data semantics. This objective can be accomplished at different levels. Database management systems clearly distinguish between two parts, the structure of the database schema and the instances contained on it.

The first stages of database design are pervaded with semantics, after the schema is created most of that semantics are ignored. Semantic models contain a complex definition of the problem which can be transformed into a database schema. The Entity-Relationship model can be transformed to a logical model using a few rules. The process the other way around it is not so easy as some semantic is lost in the translation. If users want to fully understand the problem domain, they must know the semantic model, which is not stored in the database. The fact is that once the schema is created, the semantic model is almost

useless for normal users.

Ontologies depict the problem domain, its structure and its instances, together. Structure determines relationships between data, similar to semantic models. The study of similarities between semantic models and ontologies can help us to develop proper structures to store semantics of database schemas.

In order to manage semantic information at structural level, we propose the use of ontologies for the conceptual design of database schemas. Ontology instance data is stored in the database schema after it is created. The ontology must be stored in the database to provide information about the semantics of structures. This way we have different components, the database schema with the instances and the ontology itself. Additional information must be provided to connect both representations, in order to be able to determine mappings between the schema and the ontology. With all this information available in the database it is possible to do elemental reasoning such as class subsumption.

Data instances also contain its own semantics. We distinguish between the processing of numeric attributes and text attributes.

We provide more semantics to numeric attributes through querying. Using Fuzzy Logic in databases it is possible to store and query flexible representations of numeric attributes. An user can build queries using approximate values, ranges or possibility distributions. In order to express the ranges of fuzzy numeric attributes in the ontology new XML Schema types must be created.

Textual attributes in relational databases contain useful information that sometimes is not handled in a proper way. Syntactic querying of textual fields provides only superficial information, avoiding issues related to the semantics of data. In order to extract relevant information from these fields, we must process them. Not all information in a text is relevant, so we must represent a summary of the most relevant contents of the text. This intermediate representation is created using data and text mining techniques, and it is intended to be semantically extended using external knowledge sources.

In order to better structure the process of extracting a semantic representation from text, we propose a general methodology. This methodology can be tailored using a wide variety of resources in each step. The application of the methodology results in the generation of an ontology. This ontology can be seen as a semantic summarized representation of the domain of the text attribute. Each concept in the ontology is annotated with search terms which can be used to retrieve that particular concept from other texts. This new ontology could be stored in the database too, but its management is different from those defining structural semantics and used to create the schema.

Semantic extension of intermediate representations of text to ontologies can be performed using different external knowledge sources. In our case we will analyze WordNet and Wikipedia. We will discuss how to use these tools in the context of the methodology and we will evaluate the results obtained.

In summary, we study different ways to deal with semantics in relational databases. A graphical description of the whole proposal can be seen in Figure 1.1. We can deal with semantics at two levels.

- *Giving semantics to database structures*: Starting with a ontology we store it, along

with its instances in a database.

- *Obtaining a semantic interpretation of database attributes*:
  - *Numeric Attributes*: Storage and management of numeric attributes is done using fuzzy logic. This allows to create flexible queries over data.
  - *Text Attributes*: We obtain additional information from text attributes using data mining to create an intermediate representation which can be extended with semantics an presented as an ontology. This ontology can be stored in the system.

All ontologies are expressed in OWL, a Semantic Web standard.

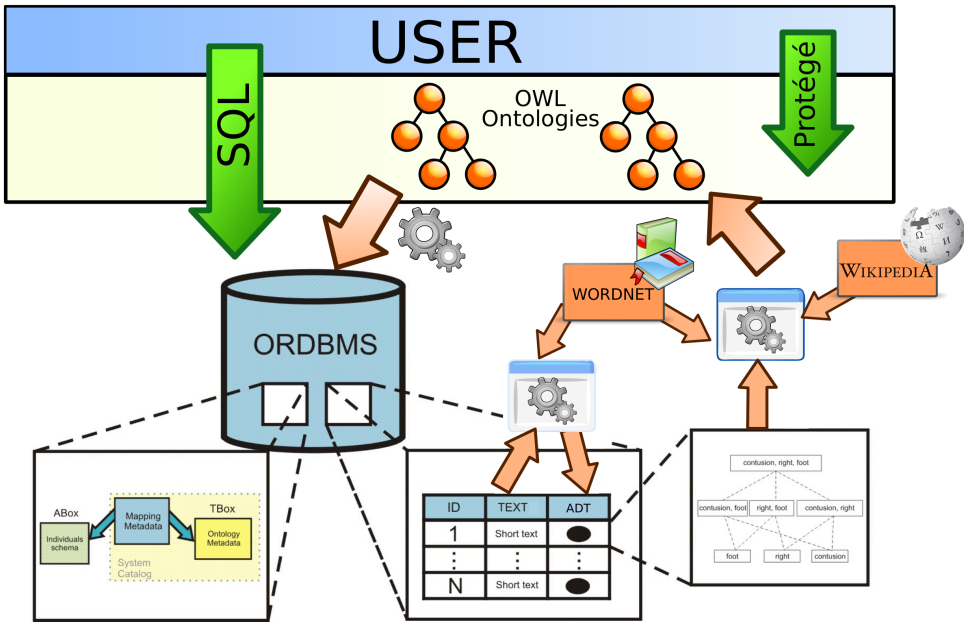


Figure 1.1: General Outline of the Proposal

The proposal is divided in two chapters. Chapter 2 describes the general process to transform OWL ontologies to database schemas. A catalog structure to store the ontology an its instances is presented. A detailed description of the transformation algorithm is given, and numeric fuzzy attributes handling is discussed. Chapter 3 presents a general methodology for the semantic extension of non-structured texts. Implementations of the methodology using WorNet and Wikipedia are presented. The role of Wikipedia in the stage of semantic extension is discussed in detail. Finally, we present some conclusions and future work.

Chapter 2

Database Schema Generation from OWL Ontologies

The Semantic Web as defined by Berners-Lee, Hendler and Lassila in [Berners-Lee et al., 2001] provides a common framework to allow data to be shared and reused across applications, enterprises, and communities. The goal of the Semantic Web is to provide the framework to allow humans and machines to use its contents in an efficient and simple way.

Ontologies have been proved to be crucial in the development of this new model of the Web. Due to their potential to describe the semantics of information and the capacity to solve heterogeneity problems, ontologies are one of the key tools in the work with semantic contents, and are also used in data integration tasks. In recent years, several standard Semantic Web languages for ontology development have emerged. Resource Description Framework (RDF) and Schema (RDFS), the Web Ontology Language (OWL) and its three sub-languages OWL Lite, OWL DL and OWL Full, and recently the new version OWL2, are the leading languages of the Semantic Web.

The use of ontologies as knowledge representation formalisms and its role in the development of the Semantic Web, motivates the creation of lots of ontologies containing huge quantities of instance data. There exists a wide variety of RDF instance data sets publicly available. All this instance data must be managed and stored. Current ontology reasoners are not capable of managing efficiently large amounts of instance data. In this context, it is a good idea to look for solutions to these problems in other well known systems.

Nowadays, Database Management Systems are the most widespread and efficient data storage solution. A vast amount of ontology based Semantic Web data is going to be stored in relational databases. Moreover, most of the data in the traditional Web is already stored in relational databases. If we want to perform a smooth transition from the current web to a Semantic Web model, we must adapt the current content and make new content suitable to be processed in a relational database. So, integration of ontology data and relational databases seems to be a clear objective for the development of the future web.

The main advantages of the use of a DBMS and features that ease the implementation of basic reasoning tasks in the database are:

- Capacity for storing and sharing large amounts of data.

- Optimized for data processing (indexes, clusters, views, etc.).
- Complex data handling features and extensibility for systems based on standard SQL:2003.
- Importation/exportation of different data formats to ease interoperability between systems.
- Recursive queries as considered in SQL:2003, that are useful for certain reasoning operations.
- Complex query optimizers.

Relational databases are specifically designed to deal with huge amounts of data, and ontologies are good knowledge representation formalisms. We propose a two-tier model where the conceptual tier is defined by an ontology, while data management is delegated upon a RDBMS, which can manage huge volumes of data and optimize queries.

Our first goal is the use of ontologies expressed in OWL as design tools for database schemas. Database schema design is a non-trivial task that requires knowledge about the domain of the problem to model, and knowledge of the conceptual model to use. Design of domain ontologies is neither a trivial task, but is compulsory in order to design Semantic Web systems. Once an ontology is defined in OWL, it is not necessary to create another conceptual model for database design i.e. Entity-Relationship model. The ontology holds all the necessary knowledge required to develop an appropriate schema for the data.

Ontology and instances must be stored in different structures in order to improve the access to instances while retaining the semantics of the ontology.

There are two main approaches to ontology storage in databases. One approach stores the complete ontology whereas the other creates a database schema based on the ontology definition to store instances. Each approximation has its virtues and its flaws, as will be explained later. In order to mitigate the drawbacks of each approach, our proposal uses a combination of both approaches to develop a combined schema to store the ontology and its instances without information loss. Some basic reasoning capacities can be added to the system through the combined use of the original ontology and the instance data in the database schema.

Transformation from OWL to a relational schema is performed using an algorithm. This algorithm has as input a domain ontology and generates a relational database schema to store instances. The ontology itself ($TBox$) is stored in database catalog tables specifically designed for this purpose. Instances ($ABox$) are stored in the schema previously built. Mapping betwen ontology concepts and instance data are created in order to perform basic semantic concordance tasks.

Ontologies are generally used to represent precise concepts. Fuzzy and incomplete data is everywhere around us, and hence also is on the Web, where the characteristics of some elements may be described as approximate values, ranges of values, upper and lower bounds, linguistic labels, etc. Due to this fact, it was only a matter of time that fuzzy representations of ontologies emerged to offer a more accurate view of certain concepts.

We propose to deal with fuzzyness at instance level, so a special kind of ontology that deals with fuzzy data as part of attributes definitions in classes is needed. We call these ontologies OWL Like ontologies because they are defined essentially in OWL, but the use of fuzzy data in datatype properties is not a standard.

The system relies in a Fuzzy Object Relation Database Management System (FORDBMS). This decision is based mainly on the possibilities the Object Relational paradigm offers, and the extensibility mechanisms that allow to create new datatypes and ways of dealing with them. Using the extension mechanisms, a Fuzzy Database is built over a classical object relational system, taking advantage of all features offered by the underlying system.

This chapter proposes a complete representation of OWL ontologies using fuzzy datatypes within a FORDBMS. The database schema is designed using the ontology, and the ontology itself and its instances are stored. Instance data can contain fuzzy datatypes.

2.1 Knowledge Bases and Databases

It is important to clarify the role that each of the components we are going to use holds in the general schema of our proposal. And also it is convenient to clearly distinguish how are we going to use those components.

The distinction between ontology and knowledge base is easy, as we can see an ontology as a particular case of a knowledge base where there is only information about the intensional part described by the domain. In our case, we will use the term knowledge base to refer to the intensional and extensional component of information, while the term ontology will be used to refer only to the intensional part of a knowledge base. In general, when we refer to knowledge bases we refer to description logic (DL) based knowledge bases.

There are many systems where databases coexist with DL. BDs take care of huge volumes of instance data, while knowledge representation systems manage intensional information in memory. DL provides a formal context similar to that provided by semantic models such as the Entity-Relationship model [Calvanese et al., 1998]. A problem modeled in DL can be checked for consistency and correction using a reasoner.

DL knowledge bases contain two distinct components, intensional knowledge in the *TBox* and extensional knowledge in the *ABox*. The *TBox* contains structural information, general properties and relations between concepts. The *ABox* contains knowledge about instances of the domain. This separation not only is done based on the different types of knowledge, it also takes into account temporal evolution criteria. Knowledge in the *TBox* do not change over time, while knowledge in the *ABox* has a temporal character and can be created, updated or deleted.

In a sense, databases also present a similar behaviour. The database schema encodes intensional knowledge about the domain of a problem, and instance data stored in tables is a sort of extensional definition of the schema. This way we can say that a database schema is like a *TBox* in a knowledge base, while data is the equivalent to the *ABox*. However, semantics in an *ABox* are not the same as those used in database instances. One important difference between DB and KB is that the former use *closed-world semantics* while the latter use *open-world semantics* [Baader and Werner, 2003]. Absence of information in

databases is interpreted as negative information: if it is not in the database it does not exists. In a $ABox$, absent information is not interpreted as non existence, but just as lack of knowledge.

Query answering in a database is not a logical reasoning, it is just the proof of a finite model, that is, the evaluation of a formula in a fixed finite model. However, querying in an $ABox$ is more complex, because requires a non trivial reasoning process where all models (sometimes an infinite number of models) must be taken into account.

Another important difference is that ontologies do not support the primary key concept. Two instances in the ontology with the same name can refer to different individuals.

These differences force us to decide which of these behaviours use. In this case, we think that a closed-world assumption for querying is the best solution, because in query answering scenarios response time is a key factor. This decision limits the reasoning tasks that can be performed, but it is essential if we want to perform query answering over instance data. In this scenario we can only do basic reasoning tasks as class subsumption.

2.2 Ontologies as Relational Database design tools

In order to fully develop the potential of ontologies and to enable efficient and flexible information gathering, persistent storage of ontologies and its retrieval is of vital importance.

There are different approaches to store ontologies. One is to build a specific purpose database system from scratch, specially designed to store and retrieve ontology data. Other approach is based on exploiting the modeling capabilities of object oriented database systems to store ontologies. Another approximation is the use of relational database management systems to store ontologies.

This last approach is particularly useful due to the widespread use of RDBMSs as data repositories on the web. A huge amount of the data present in the web is part of the Deep Web, that is, is stored in databases and it is not directly visible. So it is possible to argue that the web as it is known today relies heavily on relational databases. Additionally, ontology instance management is a complex task that is best performed by optimized data management software such as databases.

The point is that relational databases are going to play a crucial role in the development of the Semantic Web too. Probably most of the information in the Semantic Web will end up stored in relational databases. Most of the information already present in relational databases will take part of the Semantic Web by means of ontology mappings and transformations, but also new applications will be developed from scratch. This is the context on which our proposal is grounded.

Semantic Web data is data heavily relying in ontology definitions. Actually, data is comprised of ontology instances. So, when a new application is developed and it is necessary to design a database schema for ontology instances storage, a domain ontology describing application data can be used as a conceptual model for the relational database schema design.

Relationships between existent database schemas and ontologies through mappings have been widely studied in literature [Barrasa et al., 2004, Bizer and Seaborne, 2004].

But there is no such profusion of works on using ontologies as conceptual models.

In [Trinkunas and Vasilecas, 2007] an automatic transformation from ontology to the conceptual data model Entity-Relationship (ER) is presented. The graph oriented approach used, converts an ontology expressed in OWL DL into an ER model. This approach delegates the DDL and DML sentences generation to a commercial tool. The problem with this solution is that, once the schema is designed, the ontology is useless given that there is no way of establishing a correspondence between the schema and the original ontology. This contingency automatically discards the method as a possible tool for designing Semantic Web systems.

An insightful and deeply overview of domain ontologies from a database perspective is realized in [Jean et al., 2006]. A new and more complete definition of domain ontology is proposed along with a taxonomy of domain ontologies. Also, the use of ontologies as conceptual models is discussed. A domain ontology can be used as a first level of database concept specification, which later can be refined to incorporate particular requirements and hence define a conceptual model.

Several works deal with the idea of domain ontologies as conceptual models. In [Roldán García et al., 2005] a design methodology for Semantic Web database based systems is presented, and it is used to design new databases and to add semantics to existent systems. Among the different possibilities presented, we focus in that which uses an ontology as a starting point in the design of a database schema. A domain ontology is presented as an abstraction of the knowledge of the data source schemas. After a process of refinement is performed, the ontology is adapted to a local schema. The refinement process comprises two steps. In the first step, the designer extends the initial ontology providing new concepts, relationships and integrity constraints relevant to the developed system. Next, a subset of the ontology obtained in the previous step is selected and renamed in order. The new ontology obtained will be the basis of the implementation of the database schema.

The work [Sugumaran and Storey, 2006] studies the role of domain ontologies in database design thoroughly. In this case, the approach is slightly different, the ontology is used to advise the designer about the correctness and completeness of the conceptual model created by her/him using domain knowledge contained in the ontology.

In our proposal, we use OWL DL ontologies as a formalism to define conceptual models for database schema design. The design of a database schema is reduced to the appropriate design of a domain ontology that captures all requirements of data. The design of the ontology can be done by using CASE tools such as Protégé [Pro, 2008] and SWOOP [Kalyanpur et al., 2006]. Once the ontology is defined, trough a transformation process the ontology is expressed as a set of DDL and DML sentences implementing a relational database schema. An ontology expressed in OWL DL is enough to create the database schema, reducing the problem of database design to that of proper domain ontology definition.

Once the database schema is created, it is important to deal with data instances and their relation to the original ontology; in order to analyze this, next sections study different paradigms to ontology storage in database management systems.

2.3 OWL Ontology Storage in ORDBMS

This section deals with the implementation of ontology storage in ORDBMSs. We can clearly differentiate two parts in our proposal, efficient access to instance data and the preservation of the original ontology to offer basic reasoning capabilities. The proposal is oriented towards the efficient management of large volumes of instance data in a query oriented environment, where reasoning capabilities are not as important as efficient query response time.

In order to provide efficient access to instance data, we must design an appropriate relational schema for the domain of the problem. The domain of the problem is encoded in the ontology itself. The intensional description of the ontology can be translated to a relational database schema. In order to capture as much of this intensional knowledge as possible, we must represent it using an appropriate semantic model. This semantic model must be appropriate for a later implementation in a relational system. In order to perform a seamlessly migration from OWL to relational databases, some authors [Bagui, 2009] propose an intermediate transformation to an ER semantic model. This intermediate step is useful from the point of view of schema documentation, and the formalisation of the different transformations.

Transformation of the domain information contained in the ontology to a Entity-Relationship semantic model allows to deploy the model as a schema in a relational database. As a direct consequence to these transformation, the database schema storage problem is reduced to an OWL domain ontology design problem, an its transformation to a semantic model. The ER semantic model can be transformed in a relational database model without too much effort; this process is thoroughly studied in [Chen, 1976, Bagui and Earp, 2003, Navathe and Elmasri, 2007].

After the ER to database schema transformation, we can store instance data in the database tables, but we have lost every connection with the original ontology. Without the information contained in the ontology the schema generated is like any other schema generated by other means. All semantics have been lost.

In order to perform rich semantic queries, it is necessary to preserve the original ontology information. To avoid the information loss, we propose the storage of the original ontology in the database along with the instance data. Ontology storage must be performed in such a way that properties and relations are preserved, and the original structure can be recovered using queries. The intensional part of the ontology can be stored in a specific database schema that guarantees that no information is lost.

At this point we have one problem which can be solved using two different approaches, ontology transformation to a database schema, and ontology structure storage. In order to decide which is the best way to solve this problem, we review some approaches dealing with ontology storage in databases.

In [Das et al., 2004] a method to provide support to ontology-based semantic matching in a RDBMS is presented. The paper proposes a method to store an OWL DL ontology in system-defined tables of a relational database, along with the definition of a set of SQL operators for semantic matching operations and a new indexing scheme to speed up these operations. This approach stores the ontology along with its instances in system-defined

tables avoiding information loss.

In [Vysniauskas and Nemuraite, 2006] a relational database schema representation of an OWL domain ontology is presented. OWL classes and properties are transformed into tables and relations in a relational database schema where instances are stored. Some additional tables are created to maintain certain metadata regarding restrictions. This is a very good approximation but lacks a proper representation of the ontology itself, this can limit the reasoning capacity inside the database as some information is difficult to restore after the transformation.

In a similar fashion [Astrova et al., 2007] proposes rules to transform OWL ontologies to database schemas. This general rules are oriented for human consumption and need an expert to apply them. An algorithmic formulation would be more appropriate, because it can be used to discard any ambiguities in the processing. Moreover, a complete algorithmic definition can be used to implement the transformations in an automatic way.

These latter approximations deal with ontology data storage appropriately, obtaining benefits from database efficiency, but losing the original structure depicted in the ontology. The first approach covers the ontology storage problem in a more appropriate way, but lacks of an efficient access to instance data.

In our opinion it is important to preserve the ontology definition explicitly, as an external resource or inside the database, in order to perform future reasoning tasks. It is also important to maintain ontology instances structured and separated from the ontology to ease the access to them. With these considerations in mind, we propose a RDBMS hybrid schema to store ontologies and their instances, allowing a separate or combined use. A summarized version can be seen in [Barranco et al., 2007].

In order to transform the conceptual model represented by the domain ontology to a physical implementation in a relational database, we propose a OWL to database schema transformation algorithm. This algorithm takes an OWL DL ontology as input, then generates a database schema for instance data, stores the ontology (TBox Structure) in the system catalog, inserts instance data (ABox assertions) in the schema previously created and relates instance schema data with their corresponding concepts in the stored ontology.

Using this approximation we preserve ontology information in the database and create an appropriate schema for instance handling. However, both parts are disconnected. There is no way to identify which ontology class corresponds to a specific database table. In order to stablish a connection between both parts, we design a set of tables containing metadata, in order to link ontology descriptions and its instances in the database schema.

Figure 2.1 depicts the database structures used to store the ontology and its instances. The main OWL constructs are reflected in the schema, except for those concerning ontology versioning that will be treated in future work.

The proposed schema is divided in two main parts:

- Individuals Schema: This schema is created by the transformation algorithm to store ontology individuals.

- System Catalog Ontology Tables: Store the ontology and the relationships between individuals and the ontology.

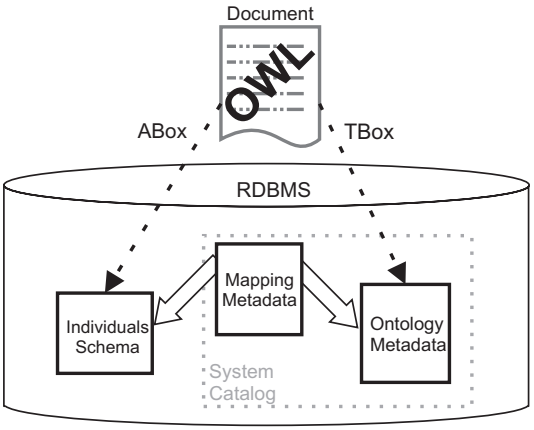


Figure 2.1: Ontology and individuals Schema

The Individuals Schema is the resulting schema of transforming an OWL DL document into a RDBMS Schema. Each different OWL DL document would create a different schema depending on the domain ontology described in it. In the database schema generation process the OWL ontology is transformed to a semantic model, ER in our case. Then, this is transformed into a database schema using known transformation rules [Date, 1990]. This way the whole process allows to transform classes into tables, object properties into relations, datatype properties into table attributes and so on. The algorithm is depicted in more detail in a later section.

Table names and attribute names are automatically generated using the name of the class with a prefix with the name of the ontology the class belongs to.

The System Catalog of Ontology Tables contains two different structures:

- Ontology Metadata: This set of tables stores the different elements of the ontology, terms, properties, relationships, restrictions...
- Mapping Metadata: These tables store the necessary data to establish a link between an individual in the Individuals Schema and the concept in the ontology to which the individual belongs.

Next sections detail each of the components of the proposed schema and define the necessary transformations from OWL ontology to relational database schema.

2.3.1 From Ontology to Semantic Model

This section deals with the formal definition of the transformation of OWL ontologies to the ER semantic model. We review the transformations necessary to express the ontology as an ER model, which will be used to build a relational schema to store instance data for the ontology.

The Entity-Relationship model was introduced in [Chen, 1976], although later other authors have contributed with some variations and extensions [Teorey, 1989, Batini et al., 1992,

Thalheim, 1993]. The ER model is the most used semantic model and it has become a standard for database conceptual design.

In order to better characterise the relation between ER and DLs we employ the formal description defined in [Borgida et al., 2003].

An ER schema $\mathcal{S}$ is built starting from pairwise disjoint sets of entity symbols, relationship symbols, role symbols, attribute symbols and domain symbols.

$$\mathcal{S}\langle E, R, U, A, D\rangle$$

Each domain symbol D has an associated predefined basic domain $D^{\mathcal{B}_\mathcal{D}}$ and it is assumed that basic domains are pairwise disjoint. A symbol with arity n has associated n role symbols, each with their respective entity symbol, and defines a relationship between those entities. We assume that each role belongs to just one relation, in such a way that determine an unique entity. Cardinality restrictions are represented by two functions applied on role symbols: $cmin_\mathcal{S}$ which returns a non negative integer and $cmax_\mathcal{S}$ which returns a value included in the set of natural positive values union the special symbol ∞. Specialization relationships between entities are modeled using a binary relation $\preceq_\mathcal{S}$.

The semantic of an ER schema can be expressed providing valid database states consistent with the schema information. A database state, formally $\mathcal{B}$ of an ER schema $\mathcal{S}$ is composed by a non empty finite set $\Delta^\mathcal{B}$, that it is assumed to be disjoint to all basic domains $\mathcal{B}_\mathcal{D}$ an a function $\cdot^\mathcal{B}$ mapping:

- every domain symbol D with their corresponding basic domain $D^{\mathcal{B}_\mathcal{D}}$,
- every entity E with a subset $E^\mathcal{B}$ of $\Delta^\mathcal{B}$,
- every attribute A with a set $A^\mathcal{B} \subseteq \Delta^\mathcal{B} \times \bigcup_{D \in \mathcal{D}_\mathcal{S}} D^{\mathcal{B}_\mathcal{D}}$, and
- every relationship R to a set $R^\mathcal{B}$ of tuples labeled over $\Delta^\mathcal{B}$.

A labeled tuple over a domain $\Delta^\mathcal{B}$ is a function of a set of roles in $\Delta^\mathcal{B}$. The labeled tuple T mapping the role U_i to an object o_i for $i \in \{1, \ldots, n\}$ is noted as $\langle U_1 : o_1, \ldots, U_n : o_n\rangle$. It is also possible to write $T[U_i]$ to denote o_i and it is named as U_i-component of T. Elements $E^\mathcal{B}$, $A^\mathcal{B}$, $R^\mathcal{B}$ are instances of E, A and R.

A database state is valid if satisfies all integrity constraints which are part of the schema. A legal database state is defined as:

A database state $\mathcal{B}$ is valid for an ER schema $\mathcal{S}$, if satisfies the following conditions:

- Each pair of entities E_1, E_2 with $E_1 \preceq_\mathcal{S} E_2$, verifies $E_1^\mathcal{B} \subseteq E_2^\mathcal{B}$.
- For each entity E, if E has an attribute A with domain D, then for each instance $e \in E^\mathcal{B}$ there is exactly one element $a \in A^\mathcal{B}$ with e as first component, and the second component of a is an element of $D^{\mathcal{B}_\mathcal{D}}$.
- For each relationship R of arity n between entities $E_1, \ldots, E_n$, for which R is connected by the roles $U_1, \ldots, U_n$, all instances of R are in the form of $\langle U_1 : e_1, \ldots, U_n : e_n\rangle$, where $e_i \in E_i^\mathcal{B}, i \in \{1, \ldots, n\}$.

- Each role U of the relationship R associated to entity E, and each instance e of E, verifies $cmin_{\mathcal{S}}(U) \leq |\{r \in R^{\mathcal{B}} | r[U] = e\}| \leq cmax_{\mathcal{S}}(U)$. A role cardinality (the number of tuples of the role in the relationship), must be greater or equal to the minimum cardinality of the role and less or equal to the maximum cardinality.

As established previously an ontology is an intensional definition, the ER schema is appropriate to represent this type of information, but it has no capacity to represent extensional information whatsoever. So, the relationship between an ontology an a schema is defined via an application.

We define the application of an ontology into an ER schema.

$$f : \mathcal{O} \longrightarrow \mathcal{S}$$

or

$$f : \mathcal{O}\langle\mathcal{T}\rangle \longrightarrow \mathcal{S}\langle E, R, U, A, D\rangle$$

Classes Classes defined in the ontology are represented as entities in the ER model. A concept C_i maps to an ER model as an entity E_i.

$$C_i \longmapsto E_i$$

A particular case are subclasses. Subclasses in an ontology are shown in an ER model as entities that are specializations too. A concept C_i which is a sub-concept for a concept C_j maps to ER as an entity E_i which is a specialization of the general entity E_j.

$$C_i \sqsubseteq C_j \longmapsto E_i \preceq_{\mathcal{S}} E_j$$

Complex classes are translated to ER as normal entities, but we need to perform additional processing in the transformation as we will see later.

Datatype Properties Datatype properties map to attributes of the entity in their domain in the ontology with basic datatypes.

Given a role R_i with domain the concept C_i defined in $\mathcal{O}\langle\mathcal{T}\rangle$ this maps to the ER schema $\mathcal{S}$ as an attribute $A_j \in A$ associated to the entity $E_i \in E$ with a domain $D_k \in D$ whose basic predefined domain is $D^{\mathcal{B}_{\mathcal{D}}}$.

Object Properties Object properties of an OWL ontology relate two classes through a property or role. In ER this is expressed as a relationship between the entities representing in the schema the ontology classes in the range and domain of the property. Cardinality of these properties in the ontology determines participation and cardinality of roles in the relationship modeled in the ER schema.

A property P_i of an ontology $\mathcal{O}$ with range the concept C_r and domain the concept C_d, is represented in the ER schema $\mathcal{S}$ as a relationship R_i between entities E_r and E_d (obtained by mapping concepts C_r y C_d), by the role U_j.

For a given object property, if minimum cardinality in the range of the property is 0, participation is partial, and for a value of 1 or greater, participation is total.

Minimum cardinality of a property P_i denoted as $\geq nP_i$ maps to the minimum cardinality of role U_j, denoted as $cmin_{\mathcal{S}}(U_j) = n$.

Analogously if maximum cardinality of property range is 1, cardinality is 1, while if it is greater than 1, the cardinality of the relationship in the ER schema is n.

Property P_i cardinality is denoted as $\leq nP_i$ and maps to maximum cardinality of role U_j, denoted as $cmax_{\mathcal{S}}(U_j) = n$.

Different combinations of participation and cardinality generate relationships $1:1$, $1:n$, $n:n$. Attending to the roles in each relationship an their cardinalities we can perform the translation to the relational model.

These mappings follow the general guidelines to OWL to ER mapping proposed by OMG with the Ontology Definition Metamodel in [OMD, 2005] although the proposal is not part of the final specification. Taking as a template this guidelines we can perform the transformation from OWL to *SQL* sentences. Table 2.1 presents these guidelines according to the most recent version of the draft including OWL to ER mappings.

Table 2.1: OWL to ER mapping

OWL Model	**ER Model**
RDFSResource	NamedElement
OWLOntology	Model
OWLClass	Entity
OWLRestriction	See OWLDatatypeProperty and OWLObjectProperty
OWLDatatypeProperty	Attribute
OWLObjectProperty	Relationship
OWLDataRange	AtomicDomain

2.3.2 OWL to Relational Schema Algorithm

After the definition of the OWL domain ontology is finished, it is necessary to define a proper database schema for instance handling. The generation of this schema is performed using a transformation algorithm which takes the OWL ontology, transform it to the ER semantic model and finally creates the schema structure in a relational database.

The individuals schema presented earlier, is the result of the application of the transformation algorithm to the OWL ontology. Table and relation definitions are created in memory and after the whole process is ended are translated to *SQL* sentences.

The algorithm is presented in pseudocode in order to show the transformation process in detail. Although the syntax employed is not very complex, we describe some details before moving to the listings.

Element `list` is a FIFO list where elements are included. Using the `NEXT` command on a list, the next object is recovered. `DBSchema` is a complex element containing table objects, it represents the relational schema to create. The behaviour of assignment operator `<-`, varies depending on if it is used before an `ADD` command or a `CREATE` command. In the first case it adds a new object, in the latter it creates it. Attributes of an element are referenced using dot syntax `element.attribute`. Command `GET element id FROM complex_element` searches for an object of type `element` with the specified `id`, inside a complex object. It can be used to retrieve tables from complex element `DBSchema`, using their identifiers.

The algorithm is presented as a succession of steps. First step is the creation of tables. Starting from the root class, each time a class is processed all its subclasses are added to the processing list. Ontology classes are transformed into tables, establishing relations according to inheritance properties such as `rdfs:subClassOf`. Subclasses generate new tables referencing the tables obtained from their super-classes. Primary key (PK) columns are named after the class from which the table is generated. The whole process expressed in pseudocode can be seen in Algorithm 1.

Algorithm 1 Table Generation

```
list IS FIFO LIST;
list <- ADD Root Classes;

DBschema <- EMPTY;

WHILE list IS NOT EMPTY

    class <- NEXT list;
    table <- CREATE TABLE;
    table.name <- class.rdf:ID;

    IF class IS ROOT THEN
        column <- CREATE PK COLUMN;
        column.name <- class.rdf:ID;
        table <- ADD column;
    ELSE
        column <- CREATE PK COLUMN;
        column.name <- class.rdf:ID;
        parentTable <- GET TABLE class.parentClass.rdf:ID FROM DBschema;
        column <- ADD FK REFERENCE TO parentTable.PK;
        table <- ADD column;
    END IF;

    IF class HAS SUBCLASS THEN
        IF class.subClass IS owl:disjointWith ONE OR MORE SIBLINGS THEN
            list <- ADD subClass;
        END IF;
    END IF;

    DBschema <- ADD table;

END WHILE;
```

Next step is the transformation of `owl:ObjectProperty` to represent relations between

the already defined tables. Depending on factors such as cardinality, new tables or attributes are added to the schema. As a general rule multi-valued object properties describing many-to-many relations are mapped as new tables with foreign keys (FK) to the `rdfs:domain` and `rdfs:range` tables with both identifiers as the primary key of the relation, whereas single-valued object properties are mapped in the `rdfs:domain` table as foreign keys to the `rdfs:range` table. There are some exceptions to this rule, as can be seen in the Algorithm 2 specification.

Datatype properties are transformed to table attributes. The attribute type is the same as the one used in `rdfs:range` in the property `owl:DatatypeProperty`. This types are valid basic XML Schema types. These types map to *SQL* types in a straightforward way. XML serialization for OWL uses XML Schema (XSD) datatypes.

Tables 2.2, 2.3, 2.4, 2.5, 2.6 show the mapping between XSD types and SQL types.

Table 2.2: XML Schema string types mapping to SQL

XSD Type	Size	Default SQL Type	SQL Compatible Type
string	n	VARCHAR2(n) if n < 4000 else VARCHAR2(4000)	CHAR, CLOB
string	-	VARCHAR2(4000) if mapUnboundedStringToLob = "false", CLOB	CHAR, CLOB

Table 2.3: XML Schema binary types (hexBinary/base64Binary) mapping to SQL

XSD Binary Type	Size	Default SQL Type	SQL Compatible Type
hexBinary, base64Binary	n	RAW(n) if n < 2000, else RAW(2000)	RAW, BLOB
hexBinary, base64Binary	-	RAW(2000) if mapUnboundedStringToLob = "false", BLOB	RAW, BLOB

Algorithm 3 shows pseudocode for the transformation of OWL datatype properties to attributes in a relationa schema table. The XSD to SQL mapping tables shown are used by function `SQLTYPE()` to determine the appropriate SQL type for a XSD type defined in the XML serialization of the OWL ontology.

Datatype properties are transformed into table attributes. The type of the columns created by `owl:DatatypeProperty` mappings concords with the datatype described in `rdfs:range`. This datatype is a valid XML XSD datatype, all these datatypes can be mapped to *SQL* types easily.

Property restrictions are a special kind of class descriptions. They describe an anonymous class, namely a class of all individuals that satisfy the restriction. OWL distinguishes two kinds of property restrictions: value constraints and cardinality constraints. Cardinality constraints have been treated already, they are covered in the mapping of `owl:ObjectProperty` properties. Value constraints in OWL are defined with the constructs `owl:someValuesFrom`, `owl:allValuesFrom` and `owl:hasValue`.

The `owl:allValuesFrom` restriction requires that for every instance of the class that has instances of the specified property, the values of the property are all members of the

Algorithm 2 Relation Generation

```
list IS FIFO LIST;
list <- ADD ALL Root owl:ObjectProperty;
DBschema HAS CONTENT;

WHILE list IS NOT EMPTY
    property <- NEXT list;
    domain <- property.rdfs:domain;

    IF domain.owl:cardinality == 1 OR domain.owl:maxCardinality == 1 THEN
        IF property IS SUBPROPERTY THEN
            column <- CREATE COLUMN;
            column.name <- property.rdf:ID;

            IF property.rdf:type.rdf:resource IS owl:TransitiveProperty THEN
                column <- ADD UNIQUE CONSTRAINT;
            END IF;

            IF property.owl:cardinality == 1 OR
               property.owl:minCardinality == 1 THEN
                   column <- ADD NOT NULL CONSTRAINT;
            END IF;

            tableParentRange <- GET TABLE property.parentProperty.rdfs:range.rdf:resource
                                FROM DBschema;

            column <- ADD FK REFERENCE TO tableParentRange.FK
            tableRange <- GET TABLE property.rdfs:range.rdf:resource FROM DBschema;
            tableRange <- ADD column;
            DBschema <- UPDATE tableRange;
        ELSE
            column <- CREATE COLUMN;
            column.name <- property.rdf:ID;

            IF property.rdf:type.rdf:resource IS owl:TransitiveProperty THEN
                column <- ADD UNIQUE CONSTRAINT;
            END IF;

            IF property.owl:cardinality == 1 OR property.owl:minCardinality THEN
                column <- ADD NOT NULL CONSTRAINT;
            END IF;

            IF property.rdf:type.rdf:resource IS owl:TransitiveProperty THEN
                column <- ADD UNIQUE CONSTRAINT;
            ENd IF;

            tableRange <- GET TABLE property.rdfs:range.rdf:resource FROM DBschema;
            column <- ADD FK REFERENCE TO tableRange.FK
            tableDomain <- GET TABLE property.rdfs:domain.rdf:resource FROM DBschema;
            tableDomain <- ADD column;
            DBschema <- UPDATE tableDomain;
        END IF;
    ELSE
        IF property IS SUBPROPERTY THEN

            table <- CREATE TABLE;
            table.name <- property.rdf:ID;
            columnParentRange <- CREATE PK COLUMN;
            tableParentRange <- GET TABLE property.parentProperty.rdfs:range.rdf:resource
                                FROM DBschema;

            columnParentRange.name <- tableParentRange.FK.name;
            columnParentRange <- ADD FK REFERENCE TO tableParentRange.FK;
            columnRange <- CREATE PK COLUMN;
            tableRange <- GET TABLE property.rdfs:range.rdf:resource FROM DBschema;

            columnRange.name <- tableRange.FK.name;
            columnRange <- ADD FK REFERENCE TO tableRange.FK;
            table <- ADD columnParentRange;
            table <- ADD columnRange;
            DBschema <- ADD table;
        ELSE
            table <- CREATE TABLE;
            table.name <- property.rdf:ID;
            columnDomain <- CREATE PK COLUMN;
            tableDomain <- GET TABLE property.rdfs:domain.rdf:resource FROM DBschema;

            columnDomain.name <- tableDomain.FK.name;
            columnDomain <- ADD FK REFERENCE TO tableDomain.PK;
            columnRange <- CREATE PK COLUMN;
            tableRange <- GET TABLE property.rdfs:range.rdf:resource FROM DBschema;

            columnRange.name <- tableRange.FK.name;
            columnRange <- ADD FK REFERENCE TO tableRange.PK;
            table <- ADD columnDomain;
            table <- ADD columnRange;
            DBschema <- ADD table;
        END IF;
    END IF;

    IF property HAS SUBPROPERTIES THEN
        list <- ADD subProperties;
    END IF;
END WHILE;
```

Table 2.4: Numeric Values Mapping from XML Schema to SQL

XSD Type	**Default SQL Type**	**Compatible SQL Types**
float	NUMBER	FLOAT, DOUBLE, BINARY_FLOAT
double	NUMBER	FLOAT, DOUBLE, BINARY_DOUBLE
decimal	NUMBER	FLOAT, DOUBLE
integer	NUMBER	NUMBER
nonNegativeInteger	NUMBER	NUMBER
positiveInteger	NUMBER	NUMBER
nonPositiveInteger	NUMBER	NUMBER
negativeInteger	NUMBER	NUMBER
long	NUMBER(20)	NUMBER
unsignedLong	NUMBER(20)	NUMBER
int	NUMBER(10)	NUMBER
unsignedInt	NUMBER(10)	NUMBER
short	NUMBER(5)	NUMBER
unsignedShort	NUMBER(5)	NUMBER
byte	NUMBER(3)	NUMBER
unsignedByte	NUMBER(3)	NUMBER

class indicated by the `owl:allValuesFrom` clause. This mapping is done adding an foreign key constraint in the column generated previously by the property, pointing to the primary key column of the class specified by `rdf:resource`.

In the case of `owl:someValuesFrom` at least one of the properties of an instance of the class must point to an individual that is part of the class indicated in the restriction. As can be seen, this scenario is much more complex because if the property is multi-valued the consistency must be checked using a trigger. If it is single-valued, the value for the property must be one of the class in the restriction, making this particular case equivalent to the `owl:allValuesFrom` previously depicted.

Also `owl:hasValue` restriction is a complex case. The property links a restriction class to a value, which can be either an individual or a data value. In the case where the value is a data value it can be modeled as a *SQL* `CHECK` constraint. Problems arise when the value is an individual, then the use of a trigger is requested. All information needed by the trigger to operate is available in the Mapping Metadata table.

Particular cases implementing triggers are obviated for now, but they are taking into account as future work. The algorithm performing the mapping of restrictions is depicted as Algorithm 4.

In the final step, instances are introduced in their corresponding tables. It is important to take into account that one instance can be separated between different tables as a result of the mapping performed. For each instance, values are stored in their corresponding columns. The database can be accessed as any other relational database, all data can be

Table 2.5: Date/Time XML Schema Types mapping to SQL

XSD Date/Time	Default SQL Type	SQL Compatible Types
dateTime	TIMESTAMP	TIMESTAMP WITH TIME ZONE, DATE
time	TIMESTAMP	TIMESTAMP WITH TIME ZONE, DATE
date	DATE	TIMESTAMP WITH TIME ZONE
gDay	DATE	TIMESTAMP WITH TIME ZONE
gMonth	DATE	TIMESTAMP WITH TIME ZONE
gYear	DATE	TIMESTAMP WITH TIME ZONE
gYearMonth	DATE	TIMESTAMP WITH TIME ZONE
gMonthDay	DATE	TIMESTAMP WITH TIME ZONE
duration	VARCHAR2(4000)	none

Algorithm 3 Attribute Generation

```
list IS FIFO LIST;
list <- ADD ALL owl:DatatypeProperty;

DBschema HAS CONTENT;

WHILE list NOT EMPTY

    property <- NEXT list;
    domainTable <- GET TABLE property.rdfs:domain.name FROM DBschema;
    column <- CREATE COLUMN WITH TYPE SQLTYPE(property.rdfs:range.rdf:resource);
    column.name <- property.rdf:ID;
    domainTable <- ADD column;

END WHILE;
```

queried and modified in the schema.

When the ontology definition is finally stored in the database, individuals and ontology are related trough the Mapping Metadata. This relation allows the creation of procedures and operators to perform limited reasoning tasks such as class subsumption. The use of these operators and procedures in queries, must be explicit.

2.3.3 Ontology Storage Schema

In this section the ontology storage database schema is depicted. We use a vertical table schema where all objects identifiers are stored and several additional tables to provide additional information regarding the object stored in the main table.

Main constructs of OWL ontologies are translated into the schema, with the exception of those related to ontology versioning.

The Ontology Metadata tables are organized as Figure 2.2 shows. The description of each table is the following:

Algorithm 4 Property Restriction Generation

```
list IS FIFO LIST;
list <- ADD Root Properties;

DBschema IS HAS CONTENT;

WHILE list NOT EMPTY

    property <- NEXT list;

    IF property HAS owl:Restriction THEN

       CASE property.restriction OF

        owl:allValuesFrom :
            tableDomain <- GET TABLE property.owl:domain FROM DBschema;
            tableRange <- GET TABLE property.owl:range FROM DBschema;
            column <- GET COLUMN property.rdf:ID FROM tableDomain;
            column <- ADD FK REFERENCE TO tableRange.PK;
            tableDomain <- UPDATE column;
            DBschema <- UPDATE tableDomain;

        owl:someValuesFrom:
            IF property.owl:cardinality == 1 OR
                        property.owl:maxCardinality ==1 THEN
                tableDomain <- GET TABLE property.owl:domain FROM DBschema;
                tableRange <- GET TABLE property.owl:range FROM DBschema;
                column <- GET COLUMN property.rdf:ID FROM tableDomain;
                column <- ADD FK REFERENCE TO tableRange.PK;
                tableDomain <- UPDATE column;
                DBschema <- UPDATE tableDomain;
            END IF;

        owl:hasValue:
            IF property.owl:hasvalue.rdfs:resource IS A VALUE
                table <- GET TABLE property.owl:domain FROM DBschema;
                column <- GET COLUMN property.rdf:ID FROM table;
                column <- ADD CHECK CONSTRAINT property.owl:hasvalue.rdfs:resource;
                table <- UPDATE column;
                DBschema <- UPDATE table;
            END IF;
    END IF;

    IF property HAS SUBPROPERTIES THEN
        list <- ADD property.subProperties;
    END IF;

END WHILE;
```

Table 2.6: Other XML Schema Types mapping to SQL

XSD Type	**Default SQL Type**	**SQL Compatible Types**
boolean	RAW(1)	VARCHAR2
language(string)	VARCHAR2(4000)	CLOB, CHAR
NMTOKEN(string)	VARCHAR2(4000)	CLOB, CHAR
NMTOKENS(string)	VARCHAR2(4000)	CLOB, CHAR
Name(string)	VARCHAR2(4000)	CLOB, CHAR
NCName(string)	VARCHAR2(4000)	CLOB, CHAR
ID	VARCHAR2(4000)	CLOB, CHAR
IDREF	VARCHAR2(4000)	CLOB, CHAR
IDREFS	VARCHAR2(4000)	CLOB, CHAR
ENTITY	VARCHAR2(4000)	CLOB, CHAR
ENTITIES	VARCHAR2(4000)	CLOB, CHAR
NOTATION	VARCHAR2(4000)	CLOB, CHAR
anyURI	VARCHAR2(4000)	CLOB, CHAR
anyType	VARCHAR2(4000)	CLOB, CHAR
anySimpleType	VARCHAR2(4000)	CLOB, CHAR
QName	XDB.XDB$QNAME	none
normalizedString	VARCHAR2(4000)	none
token	VARCHAR2(4000)	none

Table `Ont_NameSpaces` includes information about the namespaces defined and the prefixes used, a namespace can be used with different prefixes. Each term in the ontology has its corresponding namespace associated in order to deal with elements defined in different ontologies.

`Ont_Ontologies` stores general information about the different ontologies. Each ontology should also be described as a term in order to be associated with its corresponding namespace.

Table `Ont_Terms` contains the representation of the various concepts described in the ontology. Ontologies, classes and properties are stored here as basic elements to use in more complex constructs. Each term has an associate ontology and a namespace. In addition to these terms two complex elements can be defined here, `ClassList` that represents a list of classes, and `Group` which represents a group of individuals. The basic OWL constructs must be defined here before any ontology is inserted, in order to use them to create relations between terms.

`Ont_ObjectProperties` contains information about object properties and their characteristics. The domain and range of the property are stored as terms in the `Ont_Terms` table. The remaining attributes include the textual description of the property, and all

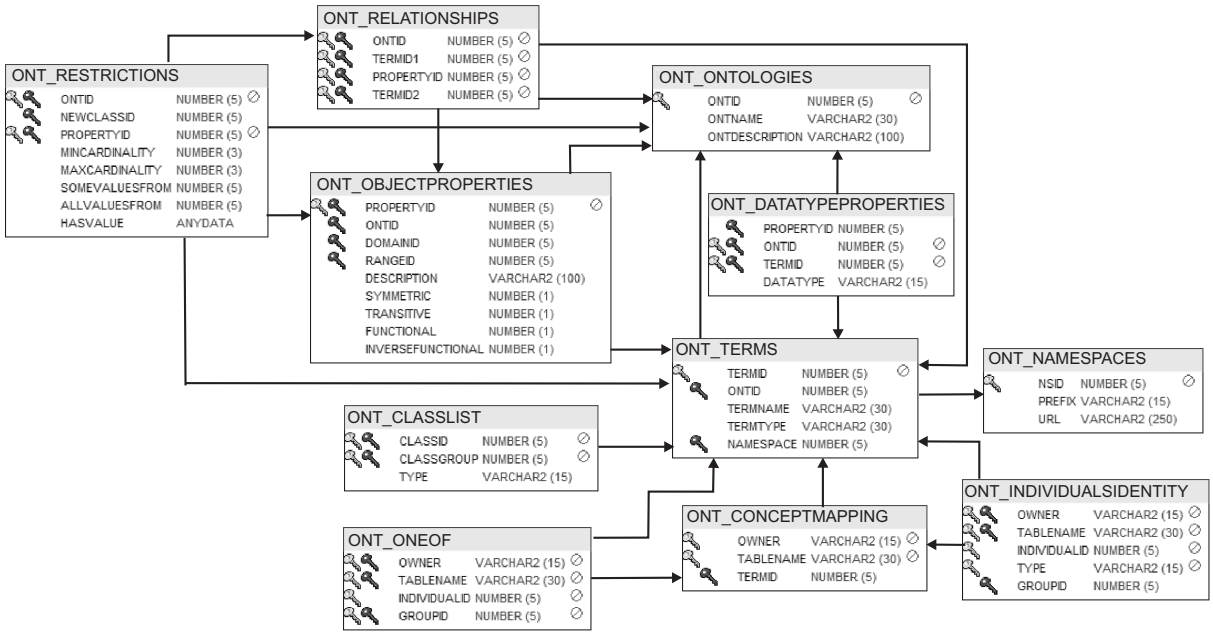


Figure 2.2: System Catalog Metadata Schema

its possible characteristics. Object properties should be included as terms in the table `Ont_Terms` in order to be used as part of relationships.

The table `Ont_Restrictions` includes information about property restrictions and the anonymous class that is created when restrictions are applied on the property. The `mincardinality` and `maxcardinality` attributes contain positive numeric values of cardinality. Attributes `somevaluesfrom` and `allvaluesfrom` contain a term identifier corresponding to a class term defined in the `Ont_Terms` table.

`Ont_Relationships` contains information about the existing relationships between two terms of the ontology. Basic OWL relationships as `SubClassOf` can be expressed here. The `termID` attributes point to terms defined in the `Ont_Terms` table. This terms can be class terms, property terms, group terms or class list terms.

`Ont_DataTypeProperties` contains information about datatype properties such as the identifier of the property, the ontology to which it belongs, the domain and the range datatype. The domain is a term in `Ont_Terms`. Datatype attribute specifies the name of the type, that could be the ones defined in the `xsd` XML Schema or fuzzy ones as we will describe later.

The Mapping Metadata Tables relate the individuals in the Individuals Schema with the Ontology Metadata tables where the proper concepts are defined. The tables are defined as follows:

`Ont_ConceptMapping` relates a table and hence its tuples (individuals) in the Individuals Schema with its corresponding class term in `Ont_Terms`. A class is represented as a table to contain its individuals, and as a term to allow reasoning.

The table `Ont_IndividualsIdentity` contains information related to issues concerning individuals identity across ontologies. OWL defines a set of properties to deal with individuals identity `owl:sameAs`, `owl:differentFrom` and `owl:AllDifferent`. These properties relate individuals, not classes. Due to this, we deal with them in a separate system-defined table which links instances and definitions in an ontology. The `type` attribute describes the OWL property, one of the three mentioned before, and `groupID` is the identifier of the group containing the individuals that are related by the property. The group of individuals definition must be included as a term in the `Ont_Terms` table.

OWL provides the `owl:oneOf` construct to specify a class via a direct enumeration of its members. `Ont_OneOf` contains information about the individuals belonging to a particular enumeration. The `owl:oneOf` construct relates a class with the individuals describing it. In order to reflect them on the system catalog it is necessary to include a relationship between the class and a group of individuals. `owl:oneOf` should be present as an available OWL property in Ontology Metadata Schema and the class and group should be inserted on the `Ont_Terms` table, then the list of individuals belonging to the group is specified in `Ont_OneOf`.

OWL also offers set operators like `owl:unionOf` and `owl:intersectionOf` to define complex classes based on operations performed on classes already defined. Classes constructed using those operators are like definitions, where their class extension consist of exactly the individuals obtained in the operation. `Ont_ClassList` contains the definition of class lists. `owl:unionOf` and `owl:intersectionOf`, should be present as OWL properties defined in the Ontology Metadata Schema, in order to create a relationship between a class and its class list definition using these properties. Once the relationship is defined, the new class is created as a view over the classes in the group stored as tables, under the selected operation.

2.3.4 Fuzzy Datatype Management

Once the general process is depicted, it is important to explain how we deal with fuzziness in our proposal. This proposal deals with fuzzy datatypes at two levels, the expression of fuzzy datatypes at ontology level and their storage and operation within a database.

Fuzzy data handling in databases is a topic widely studied, several works in literature propose different models and methods to deal with imprecision. However, although fuzzy data handling is also a hot topic in ontology research, there is not a standard for representing fuzzy data in ontologies, especially in OWL ontologies.

Works dealing with fuzziness in ontologies usually center their attention in describing fuzzy concepts and relations in ontologies, including fuzzy hierarchical relationships. In [Straccia, 2006] a fuzzy description logic is presented where concepts modeled as fuzzy

sets can be class attributes. Working with fuzziness at this level entails an increment of complexity in the representation and management of the data.

We consider that the use of fuzziness at the datatype level allows to represent certain semantics, approximate values, rages, etc, but does not affect complexity. Fuzzy data is used at instance level and it is not necessary to deal with it at ontology level. As we clearly distinguish between the treatment of the ontology structure from that of the storage and management of instance data, it is possible to add semantics to data using the capabilities provided by the underlying Fuzzy ORDBMS.

The use of fuzzy logic in queries helps the user to create meaningful queries expressing flexible criteria. In [Barranco et al., 2005b] fuzzy data querying is used to improve query results, particularly when using multiple conditions. Fuzzy queries allow the user to express her/his information requirements in an intuitive and flexible way. Query results provide additional related information that although is not what the user specifically is looking for, may be of her/his interest.

Our approach starts from an OWL ontology and adds the possibility of defining fuzzy datatypes as the range of `owl:DatatypeProperty` properties. The transformation depicted in section 2.3.2 can be easily adapted just including the fuzzy datatypes as attributes in the appropriate table in individuals schema. If the original ontology has no instance data, we just specify in the datatype property the appropriate fuzzy type for the range. These types must be defined beforehand as depicted in Figure 2.3.

If the initial ontology has instance data, it is necessary to provide the means to represent instance data using the fuzzy datatypes in the ontology. OWL ontologies do not support fuzzy types, so we must extend them. The kind of ontologies defined by the merged OWL XML Schema does not correspond with any standard ontology definition, so we will refer to these ontologies as OWL Like ontologies.

We create a new XML Schema Type named `NumericFuzzyData`. This datatype accepts values of type `CrispValue` for precise values, `TrapezoidalValue` for trapezoidal possibility distributions, `IntervalValue` for range values, `UpperBoundValue`, `LowerBoundValue` for bound values and `ApproximateValue` for triangular possibility distributions. Changing the OWL XML Schema, we allow to include these changes in the definition of datatype properties.

Now the main problem is that no reasoner is capable of working with these ontologies, because it can not recognize the new datatypes. Instead of creating a new reasoner from scratch, we propose to take advantage of the capacities offered by DBMS, in order to perform basic reasoning tasks inside the database where all ontology data is stored.

In order to better understand the capabilities of the underlying FORDBS next section presents a brief summary of its functionality.

Fuzzy Datatypes in Databases

Since the best way of dealing with vast amounts of fuzzy data is a Fuzzy DBMS, we include one in our proposal in order to store instance data with fuzzy attributes.

Among a wide variety of works on fuzzy databases [Medina et al., 1994, Umano, 1982, Pokorný and Vojtáš, 2001, Prade and Testemale, 1984, Zemankova-Leech and Kandel, 1985]

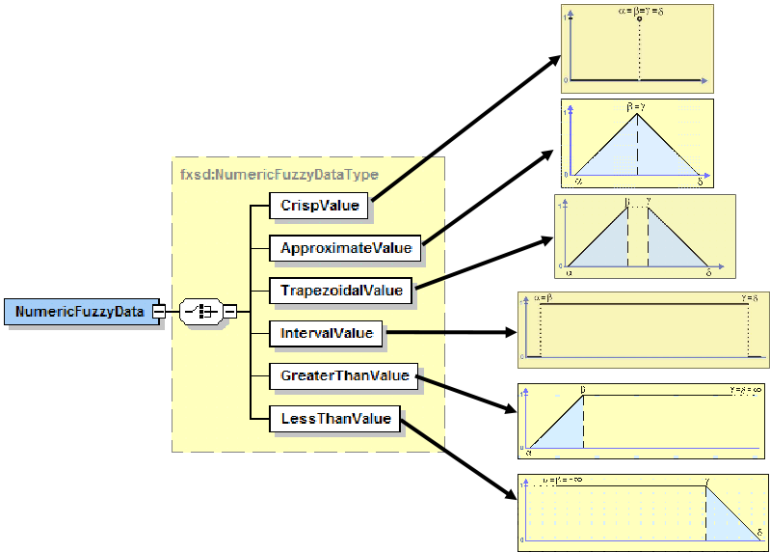


Figure 2.3: Fuzzy Datatypes

that provide different ways of handling fuzzy information, we opt for the use of a database data model [Medina et al., 2002, Cubero et al., 2004, Barranco et al., 2005a, Barranco et al., 2008b] that allows extracted fuzzy data to be stored in a FORDBMS.

This data model specifies a datatype hierarchy taking advantage of object oriented modeling benefits. Figure 2.4 shows this datatype hierarchy.

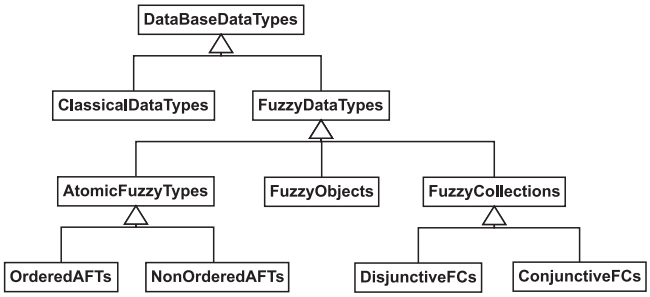


Figure 2.4: Datatype hierarchy

The FORDBMS proposal covers a wide variety of fuzzy datatypes:

- *FuzzyDataTypes* (FDT) is an abstract type which is the root ancestor of all supported fuzzy datatypes. This type declares abstract methods which must be implemented in its instantiable subtypes.

- *AtomicFuzzyTypes* (AFT) is an abstract type designed for collecting common behaviour of subtypes aimed to represent atomic data.

- *OrderedAFTs* (OAFT) datatype gives support for fuzzy numbers, which are atomic fuzzy data represented by a possibility distribution defined on an ordered domain. The order relation between domain members allows the definition of classical relational operators, from which extended relational operators are derived in order to obtain fuzzy comparators for this datatype. The extended relational operators for OAFT data are for instance FEQ (fuzzy equal to), FLEQ (fuzzy less than or equal to), etc.

- *NonOrderedAFTs* (NOAFT) datatype is designed to store fuzzy data defined on a scalar domain without any order between its elements.

- *FuzzyCollections* (FC) is an abstract datatype which extends the concept of classical collections to a fuzzy one.

- *DisjunctiveFCs* (DFC) datatype, a subtype of FC, supports fuzzy collections with disjunctive semantics.

- *ConjunctiveFuzzyCollections* (CFC) datatype is similar to DFC but supports collections with conjunctive semantics.

- *FuzzyObject* (FO) is an abstract datatype which sets a general framework for dealing with user defined complex fuzzy objects.

Initially the focus is on the use of OAFT datatypes. This includes trapezoidal distributions, approximate values, data ranges, upper bound values, lower bound values and crisp data. The creation and handling of these new numeric datatypes can have effects on efficiency. In order to solver this drawback [Barranco et al., 2008a] proposes an indexing mechanism for imprecise numerical data.

These datatypes have their counterpart in the ontology, a definition for all of them has been created in XML Schema, and is included as part of a modified OWL XML Schema.

We use OAFT datatypes because this type of fuzziness is enough to solve the kind of problems we are dealing with, where imprecision is only in the numeric attributes of data. The database model presented supports a wide variety of fuzzy types, future work will deal with these types and with concepts like fuzzy relations between objects, and fuzzy hierarchy of concepts.

Queries in a FORDMBS are processed as standard SQL queries, and can be optimized. Due to the complex nature of the fuzzy datatypes defined and the increased number of results obtained in contrast with crisp queries, processing is more complex. Nevertheless, the approach has the advantage of ranking the results and obtaining similar values to the ones searched.

2.4 Ontology Storage Example

This section presents a brief example to better illustrate the way in which data is transformed and stored using the proposed algorithm. For this example we will use the known

Wine Ontology. We show how the ontology is stored in the hybrid model. As this ontology is very big, we only show an example for each one of the OWL properties treated.

Before starting the transformation the metadata schema must be initialized with the bootstrap ontology. The elements of this ontology serve as building blocks to build complex relations. These basic elements are `owl:Thing`, `owl:Nothing` and properties such as `rdfs:SubClassOf`, `owl:IntersectionOf`, etc...

Figure 2.5 shows how the bootstrap ontology is stored, table `Ont_Ontologies` stores the name of the ontology and its identifier, similarly `Ont_Namespaces` includes the namespace, prefix and URL. Ontology terms and predefined classes are stored with the type of element they are. Properties are stored first as terms and later as properties in `Ont_ObjectProperties`. This table stores attributes such as transitivity, symmetry, etc... and contains references to the terms in `Ont_Term` which code the range and domain classes for the property.

Once the bootstrap ontology is stored it is possible to build complex structures, and relations such as inheritance,

ONT_TERMS

TERMID	ONTID	TERMNAME	TERMTYPE	NAMESPACE
1		BootStrap	Ontology	1
2	1	Thing	Class	
3	1	Nothing	Class	
4	1	SubClassOf	Property	
5	1	SubPropertyOf	Property	
6	1	disjointWith	Property	
7	1	equivalentProperty	Property	
8	1	equivalentClass	Property	
9	1	oneOf	Property	
10	1	unionOf	Property	
11	1	intersectionOf	Property	
12	1	complementOf	Property	
13	1	dataRange	DataType	

ONT_ONTOLOGIES

ONTID	ONTNAME
1	BootStrap

ONT_NAMESPACES

NSID	PREFIX	URL
1	owl	http://http://www.w3.org/2002/07/owl#

ONT_OBJECTPROPERTIES

PROPERTYID	ONTID	DOMAINID	RANGEID	SYMMETRIC	TRANSITIVE	FUNCTIONAL	INVERSEFUNCTIONAL
4	1	2	2	0	1	0	0
5	1	2	2	0	1	0	0
6	1	2	2	1	0	0	0
7	1	2	2	1	1	0	0
8	1	2	2	1	1	0	0
9	1	2	2	0	0	0	0
10	1	2	2	0	0	0	0
11	1	2	2	0	0	0	0
12	1	2	2	1	0	0	0

Figure 2.5: Bootstrap ontology storage

Next step includes classes. A basic class translates to a table in the individuals schema and an entry in table `Ont_Terms`. Figure 2.6 shows how class *Winery* is stored as a term and generates a table in the schema. Class *PotableLiquid* is subclass of *ConsumableThing*, and thus, a table is generated for each one, making *PotableLiquid* identifier foreign key to *ConsumableThing*. The inheritance relationship is reflected in the schema, but must also be reflected in the ontology metadata. In order to do this, we include an entry for each class in table `Ont_terms` and an additional entry in `Ont_Relationships` where the subclass relation is specified using property `rdfs:SubClassOf` of the bootstrap ontology.

Figure 2.7 shows how object properties are stored in the schema. The first example shows property *madeFromGrape* which is related to *Wine* and *WineGrape*. As no cardinal-

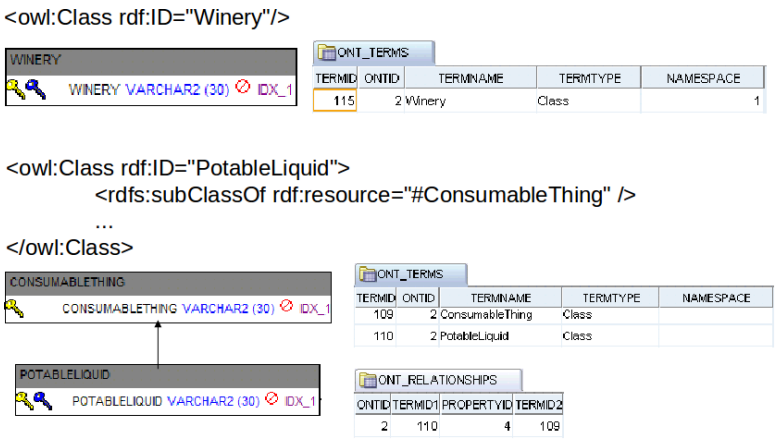


Figure 2.6: Class storage

ity constraint is given we assume a "many" cardinality (greater than 1). So the relationship is n:n and an intermediate table *madeFromGrape* which references *Wine* and *WineGrape* is created. For the ontology structure, the new property is stored as a term in `Ont_Terms`, and additional properties are described in table `Ont_ObjectProperties`.

In the second example *hasColor* is a subproperty of *hasWineDescriptor*, we create a new table which references it. The new property must be included in `Ont_ObjectProperties` with its domain identifier (the same as *hasWineDescriptor*). We must mark the property as functional in the corresponding column.

For datatype properties in Figure 2.8 we can see that attributes are added to the table representing the property domain, the attribute type is the one present in the property range. The example shows how the process is the same for basic XSD types as for fuzzy types. The property must be included in table `Ont_Terms` and additional information is stored in table `Ont_DatatypeProperty`

OWL restrictions are a special way to define classes, they define anonymous classes containing all individuals veryfying the restriction. Restrictions can be of value or cardinality. In both cases in the individuals schema these restrictions are modeled as views over data. Regarding the ontology storage in Figure 2.9, table `Ont_Restrictions` is shown. This table stores restrictions and assigns it a new class identifier.

Figure 2.10 depicts descriptions for complex classes, in particular for property `owl:unionOf`. The process to follow with `owl:intersectionOf` and `owl:complementOf` is analogous. In the schema, a view is created as the union of the tables of the corresponding classes. The list of classes involved in the operation is stored in `Ont_ClassList`, and the list identifier and the new class identifier are included in `Ont_Terms`. After that, a relation between the new class and the list of classes is included in `Ont_Relationships`. The property used in this case is `owl:unionOf` from the bootstrap ontology.

Figure 2.11 shows a more complex example where the property `owl:oneOf` allows to define a class based on the instances belonging to the class. This property must be implemented in a database schema as a trigger. With respect to the ontology structure storage,

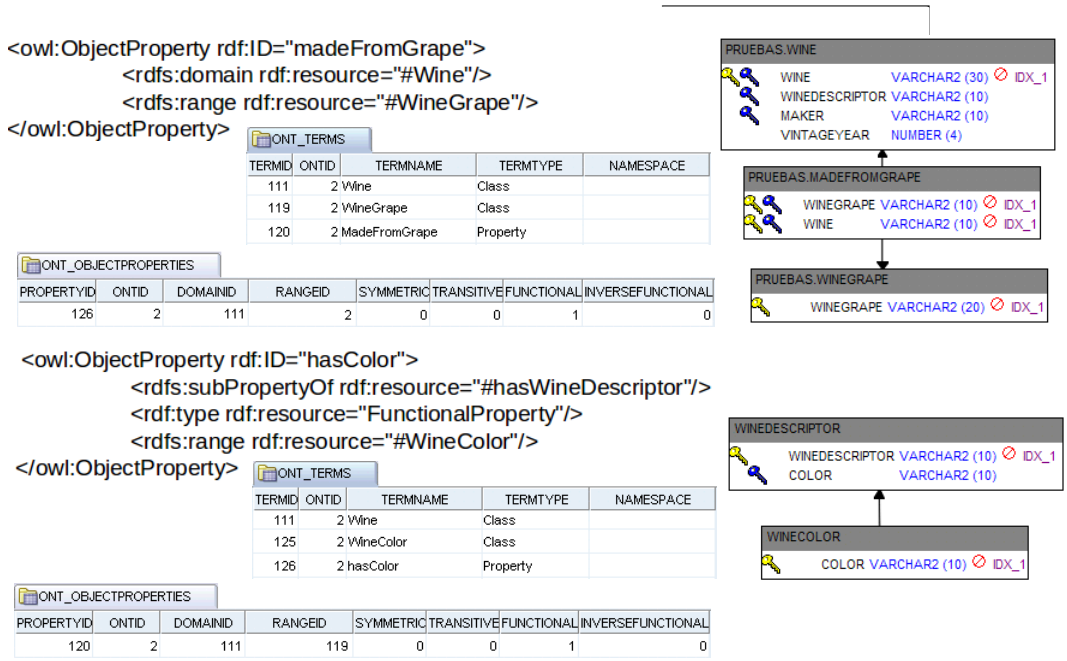


Figure 2.7: Object property storage

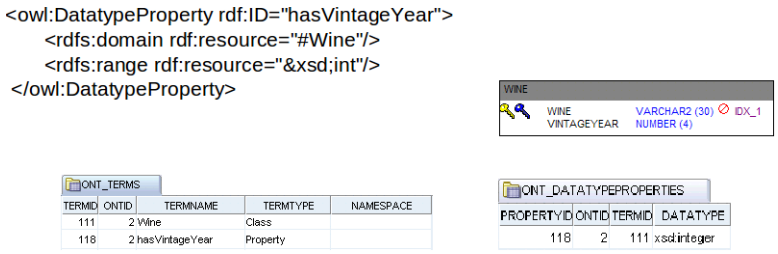


Figure 2.8: Datatype Property Storage

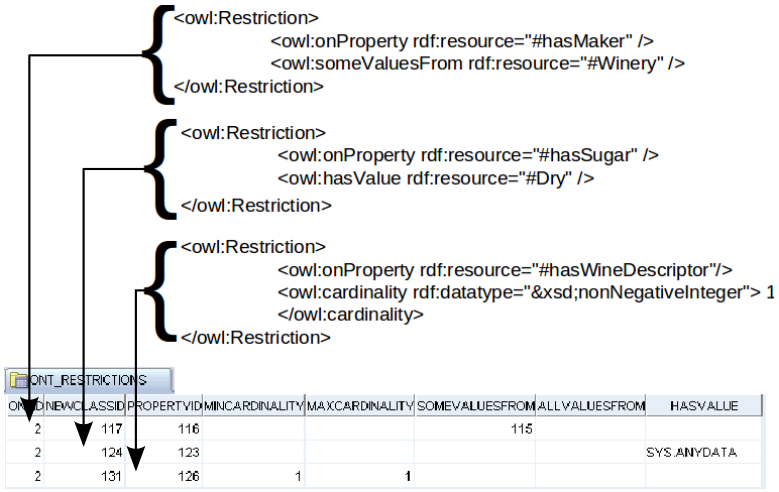


Figure 2.9: Restriction storage

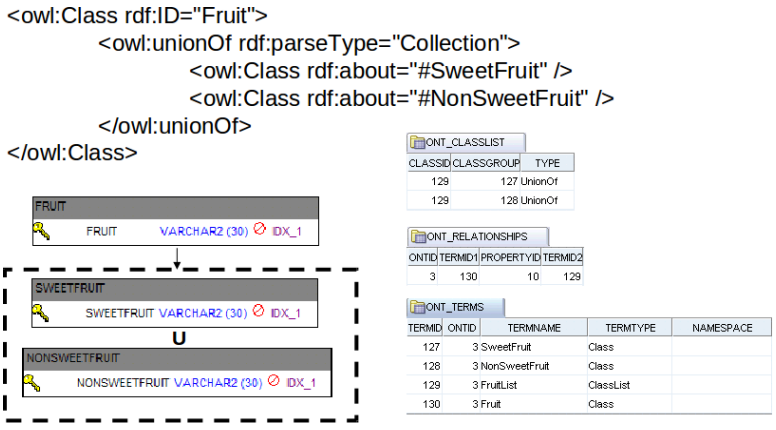


Figure 2.10: Complex class storage. Class Union

it is neccesary to define a group as a term, and later components of the group must be described in `Ont_oneOf`. A relationship stating that class *WineColor* has one of the elements in the group is added to `Ont_Relationships`.

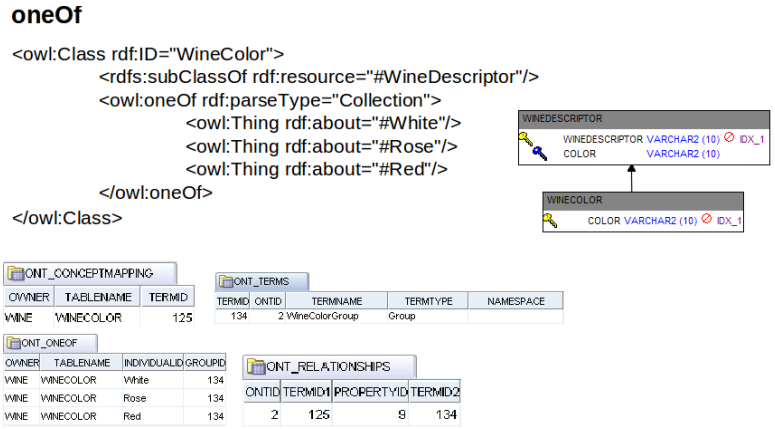


Figure 2.11: Complex class storage with `owl:oneOf`

Finally Figure 2.12 depicts information related to the management of individuals identity. This example in particular deals with property `owl:sameAs` although properties `owl:differentFrom` and `owl:AllDifferent` are treated analogously. Table `Wine` contains instances *MikesFavoriteWine* and *StGenevieveTexasWhite*, this table is associated to its corresponding term in table `Ont_Terms` through an entry in table `Ont_ConceptMappings`. The property says that they are the same, so we must create a group to include them in `Ont_IndividualsIdentity`. When instance identity is checked all information is available in `Ont_IndividualsIdentity`.

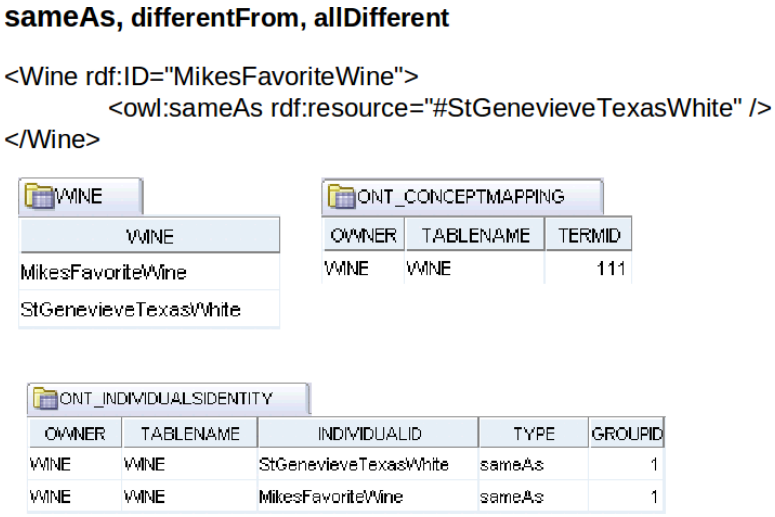


Figure 2.12: Individuals identity

As can be observed, although the process is complex, it can be easily automatized. The software prototype implementing basic features of this approach is described in Annex B of the original document. Using an OWL DL ontology, the Java program checks the ontology validity using Jena and generates SQL code. This code stores the ontology in catalog tables and generates the individuals schema.

2.5 Conclusions

In this chapter we have presented a transformation algorithm and a storage schema for OWL DL ontologies which allows to use domain ontologies as a conceptual model for database schema design. This allows to create database schemas from scratch in order to store ontology instance data in the context of the Semantic Web. The proposed schema and algorithm cover the main constructs of OWL DL ontologies and sets the groundwork necessary to develop internal reasoning in the database.

The use of an underlying FORDBMS adds the possibility to use fuzzy datatypes in the ontologies. These types provide richer semantics than traditional numerical datatypes, that can be exploited in querying. Whereas we focus on OAFT datatypes, future work will also deal with the inclusion of other datatypes defined in the FORDBMS into the ontology, the integration of fuzzy parameters at other levels, like fuzzy relations, fuzzy hierarchies, etc... analyzing the effect of these additions in the subsequent task of reasoning.

The benefit of the approach is the possibility of creating database schemas without the additional effort of designing an additional conceptual model, and the possibility to manage and query instance data in a semantic way. Data instances and the ontology can be queried together or separately, establishing a link between the objects defined and their definition, all these features supported by database technology which guarantees good performance and reliability.

The whole process of transformation into a schema and the storage in the catalog can be automatized. The software prototype developed covers some of the features described previously, except for complex scenarios which require the creation of triggers.

Future work will focus on the definition of a basic reasoner inside the database which will lead to the optimization of the schema and DML operations. Also a proper representation using triggers of complex semantic specifications is in the works.

Altough the version of OWL used is compatible with OWL2, we plan to move to the new version of the language in order to take advantage of new features.

Another concern is ontology evolution inside our system, allowing changes in the ontology and propagate them to the schema. Finally, a graphical interface to import and export ontologies into the database will be designed.

Chapter 3

Semantic Representation of Non-Structured Texts

In the previous chapter, we presented a proposal to deal with semantic data and its storage in relational databases. Information relative to the ontology structure and its instances was stored in the database. Mapping information for instances stored in database tables and their corresponding concepts in the ontology was used to connect both representations. Semantics of the ontology were preserved, and the capacities for data management of the database were put to the service of instance handling. Using extended queries we can retrieve information for a concept including all their sub-concepts. Although the semantic reasoning capabilities are reduced to class subsumption, this allows to provide additional semantics to data. But what happens to the semantics within the data?. Unstructured text attributes in databases contain useful information that cannot be accessed in a straightforward way.

Let us suppose a database containing information about scientific articles written by different research groups, and a table containing a text field with all the titles of those articles. If the user knows what he is looking for, issues a query to the database and obtains the results. Even in this ideal situation, the selection of the appropriate keywords to retrieve the requested data is not an easy task. It is even harder if the user has limited knowledge or no knowledge at all about the contents of the database. What if the user only wants to know what kind of articles are written, or the kind of topics addressed in the articles?. In a normal scenario, the user should retrieve all the titles, and then make his own guesswork.

Applying text mining techniques it is possible to obtain an intermediate representation which can be enriched with semantics extracted from electronic dictionaries and knowledge bases. The most extended formalism to represent information semantics are ontologies, so it seems the most natural way to represent semantics extracted from texts.

An ontology can provide enough information about the domain of data, and even contain the appropriate keywords to help the user build useful queries. In the context of scientific articles titles, this ontology will provide summarized contextual information about the contents of the textual field. The ontology provides the user with information

about which topics are the most frequently discussed in the articles and the appropriate keywords to retrieve those articles.

There have been different works dealing with ontological information extraction from textual sources. In [Andreasen and Bulskov, 2009] conceptual querying is used to visualize huge volumes of data. A general ontology is restricted to the set of concepts analyzed in a text. Conceptual exploration of a document set can be performed extracting the concepts from texts and providing methods to navigate and retrieve the original documents. In this work the general ontology used is WordNet. The InfoSleuth system [Woelk and Tomlinson, 1994] uses an ontology learning approach with human experts providing terms to be used as concepts in a high level ontology. In [Todirascu et al., 2001] an ontology expressed in description logics is presented. An initial ontology is built manually and a conceptual hierarchy is then built from texts. The ontology learning tool TextToOnto includes discovering of non-taxonomic relationships from texts using shallow text processing as presented in [Madche and Staab, 2000]. A comprehensive description of different approaches in ontology learning form text are discussed in [Cimiano, 2006].

In this chapter, we describe a general methodology to obtain a semantic representation from texts. The methodology proposes the transformation of texts into a structured representation of its most relevant characteristics which can be used to obtain certain semantics. The main goal is to obtain an ontology representing the domain of a text. This is achieved in three steps. First, relevant information is obtained from text using data and text mining techniques, second, an intermediate structured representation of the text is developed and third, semantic information is extracted from the structured representation to build an ontology. This methodology is a general guideline that can be implemented using several tools.

We present two different implementations of the methodology and review the tools and methods used in each step.

The first implementation is based on WordNet [Fellbaum, 1998] lexical database. WordNet is used to improve tasks such as preprocessing and semantic extension.

The second implementation is based on Wikipedia. We analyze different approaches to semantic extraction from Wikipedia, particularly focused in the use of the Wikipedia category graph and similarity and relatedness measures.

Both implementations use as intermediate representation AP-Sets [Marín et al., 2006, Martín-Bautista et al., 2006, Martín-Bautista et al., 2008], a novel technique to represent texts. The structure obtained using this technique is used as a blueprint to semantic extension using external resources.

The whole process is described in detail in the following sections.

3.1 Methodology for the Semantic Representation of Non-Structured Texts

In this section, we propose a general methodology for the semantic representation of text. This methodology is a general guideline to the process of obtaining a semantic representation from unstructured texts. Each of the steps described can be implemented using a

wide variety of tools and external resources. Depending on the tools selected, the particular implementation of some details may vary but the general structure serves as a reference.

The methodology proposal is composed of various sequential processing stages. Using well known tools and metrics, we can extract relevant information from text attributes.

The proposed stages are:

- **Syntactic Preprocessing**: This stage deals with data cleaning from a syntactic point of view, tokenization, stop word removal, etc.
- **Semantic Preprocessing**: This stage consists of the detection of synonyms, the grouping of those synonyms and the selection of a canonical representative for the synonym set. All terms in the same set are synonyms. The canonical representative of the set will replace all occurrences of the terms in the synonym set in the original text. This process allows to clearly identify the sense of a set of terms.

  In order to perform synonym detection the following tasks must be conducted:

  - *Part of Speech Tagging (POS)*: The term is tagged with the part of speech. POS is calculated in the original text according to context. The POS tags are necessary for the next step.
  - *Word Sense Disambiguation*: The sense of each term is determined. In order to ease this process information of POS is highly relevant.
  - *Synonym Set Generation*: When the sense of a term is determined, it is assigned to a synonym set. This set includes all terms with the same sense. External resources such as electronic dictionaries, synonym lists, etc, can be used to compute synonym sets and add additional terms not present in the original texts.
  - *Canonical Representative Selection*: A canonical representative is determined for each synonym set. This term acts as representative of the synonym set.
- **Intermediate Representation**: Once texts are processed the most representative features of the texts are encoded in an intermediate representation. It is desirable to select an intermediate representation able to contain additional information discovered when processing texts. The intermediate representation is going to be extended to an ontology, so it is advisable to use intermediate representations which include some type of information regarding relations between terms.
- **Semantic Extension**: Starting with the intermediate representation and using external semantic resources, relations between terms are searched. All the information encoded in the intermediate representation and that obtained from electronic dictionaries and external knowledge sources, is combined to produce a semantic representation of the texts in the form of an ontology.

The general process of semantic representation of unstructured texts is presented in Figure 3.1. This figure depicts all the stages of the methodology and points out the stages which can benefit from the use of external semantic resources.

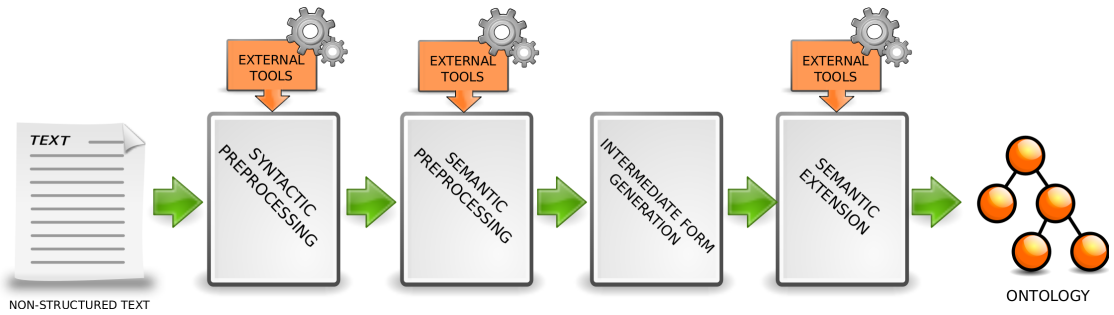


Figure 3.1: General process for the semantic representation of non-structured texts

Next section explain in detail each of the stages proposed and suggest tools that can be used in each of them. Later, we will focus on the implementation of the methodology using WordNet and Wikipedia.

3.1.1 Syntactic Preprocessing

The first step before doing a data mining or text mining process, is data preparation. In this case, we distinguish two types of data preprocessing, syntactic and semantic.

Syntactic preprocessing can be performed with any of the multitude of available preprocessing tools. Usually this stage includes tokenization, stop word removal and stemming. Depending on the data to process and the external tools to use later for semantic extension, may be advisable to skip stemming. The use of stemming algorithms such as Porter [Porter, 1980], produce stemmed forms that cannot be recognized by electronic dictionaries which produces a lot of difficulties in the semantic preprocessing.

There is a wide variety of tools which can be used to perform syntactic preprocessing in texts. Some examples for those tools are:

- **ANNIE (a Nearly-New Information Extraction System)**: It is a information extraction system developed by Hamish Cunningham, Valentin Tablan, Diana Maynard, Kalina Bontcheva, Marin Dimitrov et al. ANNIE is presented as a module inside GATE (General Architecture for Text Engineering) [Cunningham, 2002]. GATE (gate.ac.uk) is an open source project which provides solutions to text processing. ANNIE provides tools for tokenization, entity recognition (*Gazetteer*), sentence-parsing, sentence splitting using regular expressions, POS tagging, etc...

- **Lucene** : It is an open source project implemented in Java. Lucene is an API for information retrieval whose basic tasks are the indexing and retrieval of documents. As additional modules, Lucene contains utilities for text processing. Using classes derived from *Analyzer* it is possible to do tokenization, stop word removal and phrase detection for a wide variety of languages. It also provides the means to do stemming using an interface with the stemming algorithms programming language Snowball.

- **PrePro2010**: This preprocessing tool developed by Ulli Waltinger from Bielefeld University (Germany), allows to perform basic preprocessing tasks such as tokeniza-

tion, phrase detection, stemming, POS tagging, entity recognition, document structure analysis and automatic language detection.

In this stage, we can choose among a wide variety of tools but the results obtained do not present significant differences.

When this preprocessing stage is finished, texts are ready for the next stages. Depending on the intermediate representation or the way we want to process terms, we will need to performed the next stage or not. If we want to maintain the original terms to use them in a bag of words representation, we do not need to perform semantic preprocessing. However, if we are more interested in concepts than terms, semantic preprocessing helps in finding synonym terms and group them together.

3.1.2 Semantic Preprocessing

The main objective of semantic preprocessing is the syntactic homogenization of concepts present in the text. By concepts we refer to the meanings of the terms. The same concept may be represented in a text by different syntactic forms or terms. This stage intends to discover and group synonym terms referring to the same concept. We then select an unique term which represents the synonym set. This term is known as the canonical representative of the synonym set. This stage can benefit of the use of external resources such as electronic dictionaries, thesaurus and knowledge bases.

The formal definition for the selection of a canonical representative is the following:

Definition 1. *Given a set of texts $\mathcal{T} = \{T_1, \ldots, T_n\}$, and a text $T_i \in \mathcal{T}$ composed by one or more terms $t_j, j = 1, \ldots, n$. Each term t_j has one or more associated senses $s_k, k = 1, \ldots, n$ depending on the context the term is used. Given a term t_j whose context is fixed, this term can only have a disambiguated sense m_k. A disambiguated sense m_k can be the correct sense for one or more terms t_j. So, we call synonym set S_k to all terms t_j with disambiguated sense m_k.*

Definition 2. *For each synonym set S_k there is a term $t_j \in S_k$ which is called canonical representative of S_k, denoted as r_k and it is unique. Every term t_j in the synonym set S_k, can be substituted by r_k the canonical representative of the set, without changing the meaning of the original text.*

$$\forall t_j \in S_k, \exists r_k | t_j = r_k \tag{3.1}$$

Determining a synonym set is a complex task which is divided in subtasks of similar complexity. To create a synonym set we must determine the sense of all terms in the text; in order to achieve this, it is necessary to know the POS of all terms, because a term can have different meanings depending on its POS.

Let us see the tasks to perform in order to compute the synonym sets.

Part of Speech Tagging

A part of speech tagger adds a tag to each term according to the part of speech it has in a sentence context. POS tags include *noun*, *verb*, *adjective*, *adverb*, etc... . Part of speech can be determined using information previously collected about the term, or using a context structure (the relative positions of certain part of speech in sentences are clearly defined).

Tagging process is very complex, because is context dependant and some terms can change their POS when using in different contexts.

There exist different approaches in literature to the POS tagging problem, usually divided into two groups:

- **Linguistic Approaches**: Based on the gathering of rule sets provided by linguists to determine the POS using patterns.

- **Corpus Based Approaches**: These are based on machine learning techniques applied on huge data sets. Learning can be supervised or unsupervised. Supervised learning uses a tagged corpus. Using this corpus, a classifier is built; then the classifier will infer the tags of the new term introduced. Unsupervised learning uses a non tagged corpus and extract information based only on statistical values obtained from data.

There are mixed approaches that take advantage of the best features of both approaches.

We reference here some well known taggers. All these taggers are statistical ones, because they have been proved to obtain better results for this task.

- **TreeTagger** [Schmid, 1999] : Tagger based on Hidden Markov Models (HMM). Tags text with part of speech information and the stem of the term. Developed in the TC Project of the Computational Linguistic Group in the University of Stuttgart. It has been used successfully for tagging text written in German, English, French, Italian, Spanish, Greek. It can be adapted to other languages if a lexicon and a manually tagged corpus are available.

- **TnT** [Brants, 2000]: TnT, is the short name for *Trigrams'n'Tags*. This tagger is based on Hidden Markov Models an its used widespread. Developed in University des Saarlandes, TnT is very efficient and it is possible to train it for different languages and any tag set. Training is performed using an annotated corpus. This tagger can successfully deal with unknown words.

- **SVMTool** [Giménez and Márquez, 2004]: Developed in the Technical University of Catalonia (Spain), this open source tagger uses Support Vector Machines (SVM) for classification.

- **Stanford Tagger** [Toutanova et al., 2003]: Developed by the Stanford University, this tagger is one of the most precise for the English language.

When a term is POS tagged, we discard all other meanings or senses attached to other possible POSs for the term. This eases the word disambiguation process because discards non valid senses beforehand.

Word Sense Disambiguation

When terms are tagged, non valid senses are discarded, but one term can have more than one sense for a POS. With word sense disambiguation we can select the most appropriate sense for a term taking into account its context. Word senses disambiguation is a critical step in the processing of texts. A bad sense selection for a term can severely affect results.

There are several word disambiguation algorithms ranging from the most simple to those of high complexity. As the selection of the appropriate sense is a critical factor, it is important to use a good disambiguation algorithm.

Most of the disambiguation techniques rely in external knowledge sources. The disambiguation algorithms that we use are based on WordNet and thus they are reviewed in the section regarding the methodology implementation using WordNet.

Canonical Representative Selection

When all words have been disambiguated, they are assigned to a synonym set S_k according to its disambiguated sense. We define a set of synonym sets $S_k, k = 1, \ldots, n$. Each term t_j is assigned to a set S_k only. The terms in the synonym set are equivalent, any of them can be selected as canonical representative, but we choose the most frequent in the original text.

We compute frequency f for each term t_j, and select the most frequent. This term is the canonical representative r_k. All occurrences of the terms t_j in the original text are substituted by the canonical representative r_k of their sets S_k.

3.1.3 Intermediate Representation and Semantic Extension

When the text preprocessing is finished, texts are stored in an intermediate representation form. This representation can be any of the text representation forms available in the literature. The most informative the intermediate form, the better.

Once chosen the intermediate form, the texts are expressed using it. The structure containing the text is semantically extended in order to obtain an ontology.

Semantic extension is performed by using external tools that provide semantic information. The only requisite to external resources is that they must contain a conceptual taxonomy that can be exploited.

WordNet is one of the most important tools for natural language processing and can be exploited to provide terms with semantics. Recently the use of Wikipedia category graph is gaining attention as a semantic resource. There are many other sources of semantics, like the *Open Directory Project* or the medical ontology *Snomed* for medical texts processing. Any ontology that can be accessed in an automatic way, is a candidate to provide semantics in this semantic extension stage.

The semantic extension process is comprised of various tasks. The first task is to find a correspondence between the terms in the intermediate representation and the concepts in the external semantic resource used. When the appropriate concept is found, the taxonomy is traversed following a path to the root. Concepts in the path to the root are selected using a predefined criterion. The ontology is built with all the selected concepts. The resulting ontology contains the most relevant concepts included in the original text, and those closely related to them and extracted from the external semantic resource. This process can be seen as a selective prunning of a general taxonomy to adapt it to a specific domain.

3.2 Semantic Representation of Non-Structured Texts Using WordNet

The general process followed to obtain the ontology from unstructured texts is depicted in Fig. 3.1. Textual attributes are processed in order to obtain a basic ontology describing their content. To obtain the ontology, texts are processed and represented as an intermediate form using AP-Sets. The AP-Set structure obtained (AP-Structure) is then transformed into an ontology.

In this section, the methodology is adapted in order to use WordNet as a tool for semantic extension. WordNet [Fellbaum, 1998] is a lexical database containing a large amount of English word definitions and relations. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (*synsets*). Each *synset* expresses a distinct concept. *Synsets* are interlinked by means of conceptual-semantic and lexical relations.

In [Campaña et al., 2009] we presented a proposal which used WordNet to perform semantic preprocessing and to extend the intermediate representation of AP-Sets into an ontology. WordNet allows to obtain the meaning of processed textual terms. Obtaining definitions and processing all the terms in a text would produce a large and non relevant ontology. With AP-Sets representation only relevant terms to the domain are processed, thus obtaining a more compact and relevant ontology gathering the main concepts appearing in texts.

The adapted methodology using WordNet is shown in Figure 3.2

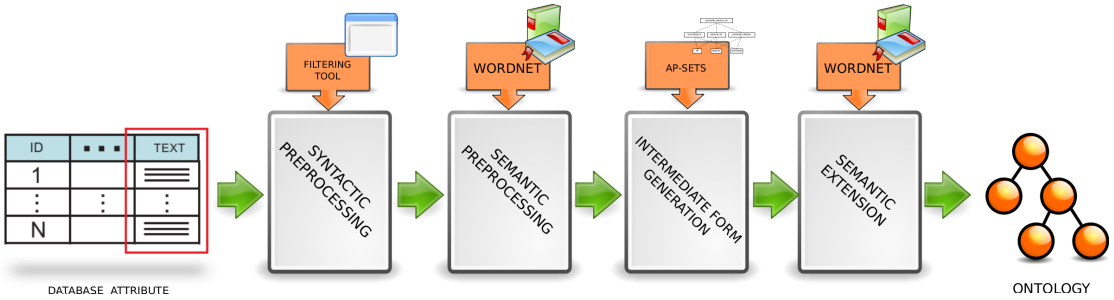


Figure 3.2: Methodology implementation using WordNet

As can be seen, the stages are the same as in the general methodology, but the tools

used to implement each of them are specified. Let us briefly present the tools used in each stage.

- Select the text attribute in the database to process.
- Syntactic preprocessing is performed using a filtering tool.
- Semantic preprocessing uses WordNet for some tasks. This stage requires:
  - POS Tagging.
  - Word sense disambiguation, using WordNet.
  - Synonym detection and grouping using WordNet.
- Creation of the intermediate representation using AP-Sets.
- Semantic extension of the intermediate representation using WordNet.

Some of these techniques and tools need further explanation. Following sections provide detailed information to fully understand them.

3.2.1 Data Preprocessing

The first step before performing a data or text mining process is the preparation of the data to obtain better results and ease of processing. Preprocessing in this case could be syntactic or semantic.

Syntactic Preprocessing

Syntactic preprocessing is performed using a tool that allows to apply filters to relational database attributes.

Using the syntactic preprocessing tool the filter applied to texts are tokenization and stop-word removal. The application of a stemmer algorithm is advised against due to WordNet inability to process stemmed word forms.

A concept can be represented by different word forms and thus not treated as the same thing when semantic extension is performed. To avoid this behaviour a semantic preprocessing using WordNet is carried out.

Semantic Preprocessing using WordNet

Semantic preprocessing is intended to detect different syntactic representations of the same concept, in order to substitute them for an unique syntactic form. This form is used as the canonical representative of the set of syntactic forms defining a given concept.

The substitution is performed using WordNet as a knowledge base. As stated above, WordNet organizes terms in *synsets* or synonyms sets. For each syntactic form, the *synset* to which it belongs, is identified and the term is replaced by the canonical representative of that *synset*.

Determining the *synset* to which a term belongs is not an easy task, it is necessary to know the part of speech of the term and its sense. The part-of-speech (POS) of a term is the grammatical category of that term in a sentence. The identification of words as nouns, verbs, adjectives, adverbs, etc, is essential to determine the correct sense of a given term. Even knowing the POS of a term, it can have many senses for that POS. Each of those senses is linked to a WordNet *synset*. In conclusion, to determine a word meaning and hence its *synset*, it is necessary to know its POS and its sense. A sense for a word can be determined through word sense disambiguation (WSD) [Agirre and Edmonds, 2007].

Part-of-Speech Tagging

POS tagging is performed using software developed by The Stanford Natural Language Processing Group. The software is a Java implementation of the log-linear part-of-speech taggers described in [Toutanova and Manning, 2000, Toutanova et al., 2003]. The tagger uses the *Penn Treebank English POS Tag Set* [Marcus et al., 1993] containing thirty-six tags for grammatical categories. WordNet only uses four gramatical categories, noun, verb, adjective and adverb; therefore a mapping between both tag sets must be done. Table 3.1 shows the description and connection between the Penn Treebank and WordNet sets. Tags with no equivalent in WordNet are ignored.

After several tests, we have found this tagger to incorrectly tag non common or technical words. This cases are tagged as proper nouns (NP o NPS). This poses a problem because they could be in WordNet but under other POS. When this happens, we try to identify all occurrences of the word in WordNet and try to determine the correct POS.

When tagging a word with its POS, we discard all senses for the word which are linked to other POSs. This makes easier to identify the correct sense in the stage of word sense disambiguation.

Word Sense Disambiguation

Disambiguation is a key task in semantic preprocessing, since an adequate *synset* must be determined for each word. A trivial disambiguation algorithm using WordNet involves the use of the most frequently used sense. WordNet senses are sorted according to the frequency of use, most frequent senses for a word are obtained first. This heuristic is of no use in semantic preprocessing, therefore different WSD algorithms are reviewed.

Lesk algorithm [Lesk, 1986] uses a set of dictionary entries, one for each sense of a word, and some knowledge about the immediate context on which disambiguation is performed. The idea behind Lesk algorithm is to simultaneously perform disambiguation through measuring overlaps between dictionary entries of the words in the context. Dictionary definition overlaps are computed as the number of words in common. As words are disambiguated simultaneously, computation time increases exponentially as the number of words to disambiguate increases.

The Simplified Lesk algorithm [Vasilescu et al., 2004], is another variant of the Lesk algorithm. It disambiguates a word each time, taking a word and computing the overlap between its dictionary definitions and the actual context. Senses with greater overlap are

Table 3.1: Penn Treebank Tag Set and WordNet correspondence

Penn Treebank	**Description**	**WordNet**
CC	Coordinating conjunction	
CD	Cardinal number	
DT	Determiner	
EX	Existential there	
FW	Foreign word	
IN	Preposition or subordinating conjunction	
JJ	Adjective	ADJECTIVE
JJR	Adjective, comparative	ADJECTIVE
JJS	Adjective, superlative	ADJECTIVE
LS	List item marker	
MD	Modal	
NN	Noun, singular or mass	NOUN
NNS	Noun, plural	NOUN
NP	Proper noun, singular	
NPS	Proper noun, plural	
PDT	Predeterminer	
POS	Possessive ending	
PP	Personal pronoun	
PP$	Possessive pronoun	
RB	Adverb	ADVERB
RBR	Adverb, comparative	ADVERB
RBS	Adverb, superlative	ADVERB
RP	Particle	
SYM	Symbol	
TO	to	
UH	Interjection	
VB	Verb, base form	VERB
VBD	Verb, past tense	VERB
VBG	Verb, gerund or present participle	VERB
VBN	Verb, past participle	VERB
VBP	Verb, non-3rd person singular present	VERB
VBZ	Verb, 3rd person singular present	VERB
WDT	Wh-determiner	
WP	Wh-pronoun	
WP$	Possessive wh-pronoun	
WRB	Wh-adverb	ADVERB

more likely to be the correct sense. Simplified Lesk improves precision and efficiency of the original Lesk algorithm, so it is a good candidate to use as a disambiguation algorithm in the semantic preprocessing. The main drawback of this approach is that overlaps are strictly based on syntactic patterns. Dictionaries like WordNet model semantic relations that can be taken into account in a disambiguation process, which lead us to the next variant of Lesk algorithm.

In [Banerjee and Pedersen, 2002] the Adapted Lesk Algorithm was introduced. While previous Lesk algorithms only considered definitions as context, the adapted algorithm uses definitions of related concepts according to the WordNet hierarchy such as hypernyms, hyponyms, holonyms, meronyms, troponyms, etc. The context is extended with the definitions of words connected by these semantic relations. This algorithm is more precise than the simplified version so it is our choice as disambiguation algorithm for semantic preprocessing.

Once words are disambiguated, each term is substituted by the canonical representative of its *synset*. This representant will be selected as the most frequent word found in the texts pertaining to that *synset*.

3.2.2 Intermediate Representation using AP-SETS

Once data is preprocessed an intermediate representation form is generated. In our case, we use the intermediate representation form of AP-Sets. This novel approach for the representation of short texts based on frequent itemset is presented in [Marín et al., 2006], where the abstract data type called AP-Set and its operations are defined. AP-Sets are an intermediate representation for texts based on the Apriori algorithm [Agrawal and R.Srikant, 1994] used in data mining.

The formal definition of AP-Sets according to [Martínez-Folgoso, 2008]:

Definition AP-Set Let be $X = \{x_1...x_n\}$ a finite set of items and $\mathcal{R} \subseteq \mathcal{P}(X)$ a set of frequent itemsets, being $P(X)$ the set of parts of X. We will say that $\mathcal{R}$ is an AP-Set if and only if:

1. $\forall Z \in \mathcal{R} \Rightarrow \mathcal{P}(Z) \subseteq \mathcal{R}$
2. $\exists Y \in \mathcal{R}$ such that :
   (a) $card(Y) = max_{Z \in \mathcal{R}}(card(Z))$ and not exists $Y' \in \mathcal{R}$ such that $card(Y') = card(Y)$
   (b) $\forall Z \in \mathcal{R}; Z \subseteq Y$

The first condition of the above definition guarantees that any AP-Set verifies the Apriori property. The second one assures us the existence of an unique set called Y of maximal cardinality *spanning set of* $\mathcal{R}$, which characterizes the AP-Set. We will denote $\mathcal{R} = g(Y)$, that is $g(Y)$ is an AP-Set with spanning set Y. Let us remark that $g(Y)$ is also the power-set of Y.

We will call *Level of* $g(Y)$ to the cardinal of Y. Obviously, AP-Set of level 1 is composed of the elements of X. We will consider the empty set $\emptyset$ as the AP-Set of level zero.

Example 1. *Let be* $X = \{1, 2, 3, ..., 10\}$ *and*
$\mathcal{R} = \{\{1\}, \{3\}, \{5\}, \{1, 3\}, \{1, 5\}, \{3, 5\}, \{1, 3, 5\}\}$, *the spanning set is* $Y = \{1, 3, 5\}$

For texts, the referential are terms and their frequencies. Any AP-Set with spanning set Y is in fact the lattice of $\mathcal{P}(Y)$. When texts are processed and frequent itemsets are computed complex structures composed of AP-Sets appear. The formal definition according to [Marín et al., 2006]:

Definition AP-Structure Let be $X = \{x_1...x_n\}$ any referential and
$S = \{A, B, ...\} \subseteq \mathcal{P}(X)$ such that:

$$\forall A, B \in S\,;\; A \not\subseteq B\,,\; B \not\subseteq A$$

We will call *AP-Structure of spanning* S, $\mathcal{T} = g(A, B, ...)$, to the set of AP-Set whose spanning sets are $A, B, ...$

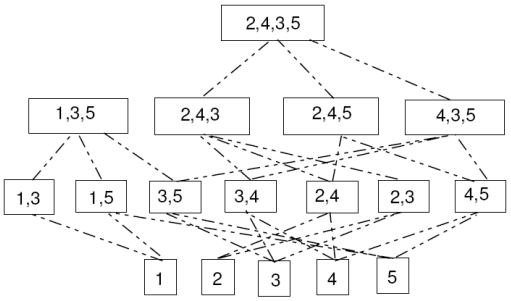


Figure 3.3: AP-Structure lattice

Any AP-Structure is a lattice of subsets whose upper extremes are their spanning sets. Figure 3.3 shows an example of the lattice underlying in $g(\{1, 3, 5\}, \{2, 4, 3, 5\})$.

In [Marín et al., 2006, Martín-Bautista et al., 2006, Martín-Bautista et al., 2008] additional formal definitions for AP-Sets operations are presented.

The mechanism to get the AP-Sets from the original textual data in order to form the AP-Structure is described as follows.

Let be R a relation with attributes $\{A_1, A_2, A_T, ..., A_n\}$, where A_T is a textual attribute without a predictable structure. To obtain the AP-Structure the tasks performed are the following [Martín-Bautista et al., 2006]:

1. Obtain a data dictionary consisting in a list of the terms removing stop-words.

2. Transform the data into a transactional database, where the attributes are the different terms of the data dictionary, and each tuple corresponds to a record.

3. Obtain the frequent itemsets of the transactional database following an Apriori like algorithm [Agrawal and R.Srikant, 1994].

4. The maximal itemsets (AP-Sets) form a lattice structure, that is, the AP-Structure, following the Apriori property.

When building AP-Sets two different structures are obtained. On the one hand a domain AP-Structure describing the domain of the column containing texts is obtained, on the other hand a new column is appended to the original relation. This new column contains an AP-Structure for each row representing the corresponding text field. The latter AP-Structure is the sub-AP-Structure induced by the text terms from the domain AP-Structure.

We are going to use the domain AP-Structure as basis for the ontology, because it contains all the relevant information extracted from the textual attribute.

Using AP-Sets as intermediate form, relevant domain information is extracted using word frequency as criteria. The domain AP-Structure is enriched with semantic relations to obtain an ontology. This process is explained in detail in a later section.

3.2.3 Knowledge Structure Generation

When the preprocessing stage is finished, it is time to generate AP-Set structures. Thanks to semantic preprocessing, frequency is increased for concepts represented by many different syntactic forms.

Following the procedure described in the previous section, AP-Sets are generated and the domain AP-Structure is created. As stated before, we focus on the domain AP-Structure, because it holds the summarized knowledge concerning the domain of a database textual attribute.

To illustrate the whole process we present a small example created over a reduced set of texts to ease interpretability. The experimental set is composed of a selection of 62 paper titles written by members of the Intelligent Databases and Information Systems research group from the *University of Granada*, *Spain*.

The configuration parameters of the Apriori algorithm are: N for the number of iterations, 0.03 for the minimum support threshold and 0.001 for the minimum confidence threshold.

With this configuration, the AP-Structure obtained is the one shown in Fig. 3.4. For the sake of brevity the AP-Structure is represented here by its maximal itemsets. As can be observed, the maximum length of the frequent itemsets found is of 3 terms.

Due to the nature of the generation process, itemsets lack of internal order. Future work will deal with a representation and generation procedure taking into account partial ordering inside itemsets. To overcome this drawback, the terms are POS tagged in order to calculate a suitable ordering as we will see later. The POS tag assigned to the term and translated into a valid WordNet POS tag is shown between brackets, where [A] stands for adjective, [N] stands for noun, [V] stands for verb and [ADV] stands for adverb. This POS tags are those of WordNet as for the type of processing performed it is not necessary to include additional tags.

The domain AP-Structure has a lattice structure, where the first level is devoted to individual terms, and upper levels contain frequent groupings of terms (level 2 contains

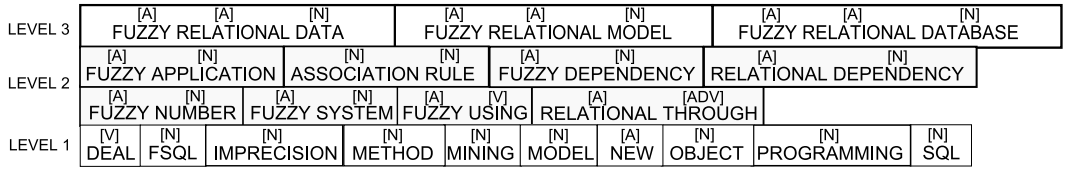


Figure 3.4: Maximal itemsets defining the AP-Structure

pairs, level 3 triplets, and so on...). Each of the terms of a frequent itemset and the combinations between them are also frequent.

The complete lattice is presented in Figure 3.5, due to its big size and in order to ease visualization, a summarized version is presented in Figures 3.6 and 3.7. In the figures, generator maximal itemsets are depicted using solid background color. To complete the lattice structure two additional nodes are included, the empty node in the lower part of the figure and the node containing all terms in the top of the figure.

As can be seen the term `MODEL` appears twice, and in both cases its part of speech is identified as `Noun`. Although at first glance this may appear as an error, it is just a side effect of semantic preprocessing. Both occurrences of the terms correspond to different senses. If both were the same, `MODEL(2)` would not be a maximal itemset, because it would be included in the maximal itemset `FUZZY RELATIONAL MODEL`. Thus semantic preprocessing has assigned two distinct senses to the same term resulting in two different itemsets `MODEL` and `MODEL(2)`. This is the reason why semantic preprocessing it is so important to provide semantics to texts.

Let use briefly analyze the most interesting parts of the lattice. It is curious to observe that although very frequent, some itemsets lack of meaning. This is particularly notorious on those composed by verbs and adjectives. This phenomenon can be explained analyzing the origin of the processed data. The original data was composed of research paper articles and those have a particular and repetitive structure. We observe that the term `NEW`, an adjective, it is of common use to emphasize the novelty of a research proposal. Something similar occurs with the verb `DEAL`, because it is usual to present a proposal *"to deal with"* something. The syntactic preprocessing got rid of the other particles and the verb *deal* made it to the AP-Structure. The same occurs with `FUZZY USING` and `RELATIONAL THROUGH`, *use* and *through* are respectively a verb and an adverb very commonly used to summarize the purpose of a research article in its title. In this case the research activity of the group from which we took the paper titles, is focused in fuzzy logic (*fuzzy logic, fuzzy numbers, fuzzy databases*) and in databases (*relational data, relational data model, relational database*). The structure of the titles combined with the research topics of the group can produce odd itemsets like the ones obtained here. We can see that adjectives, verbs or adverbs, if not associated with a name do not provide relevant information. When processing the lattice we must take into account this effect to avoid noisy terms with no relevant semantics. Later in the chapter we propose a simple method to detect and discard this type of itemsets.

In order to reduce complexity in future examples we select a subset of the whole lattice,

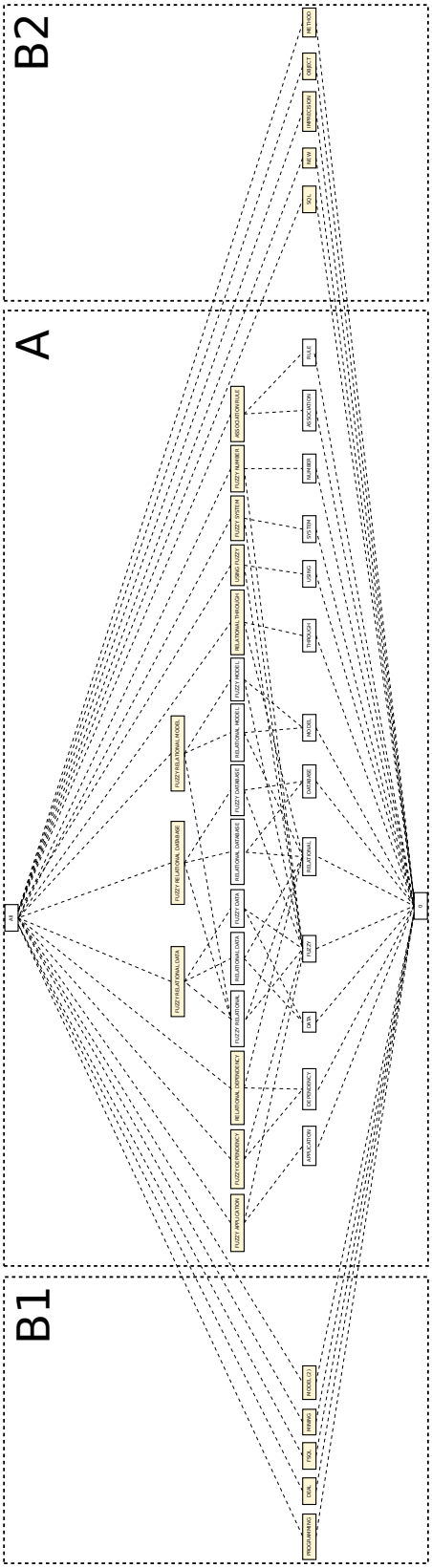


Figure 3.5: Complete lattice defined by the AP-Structure

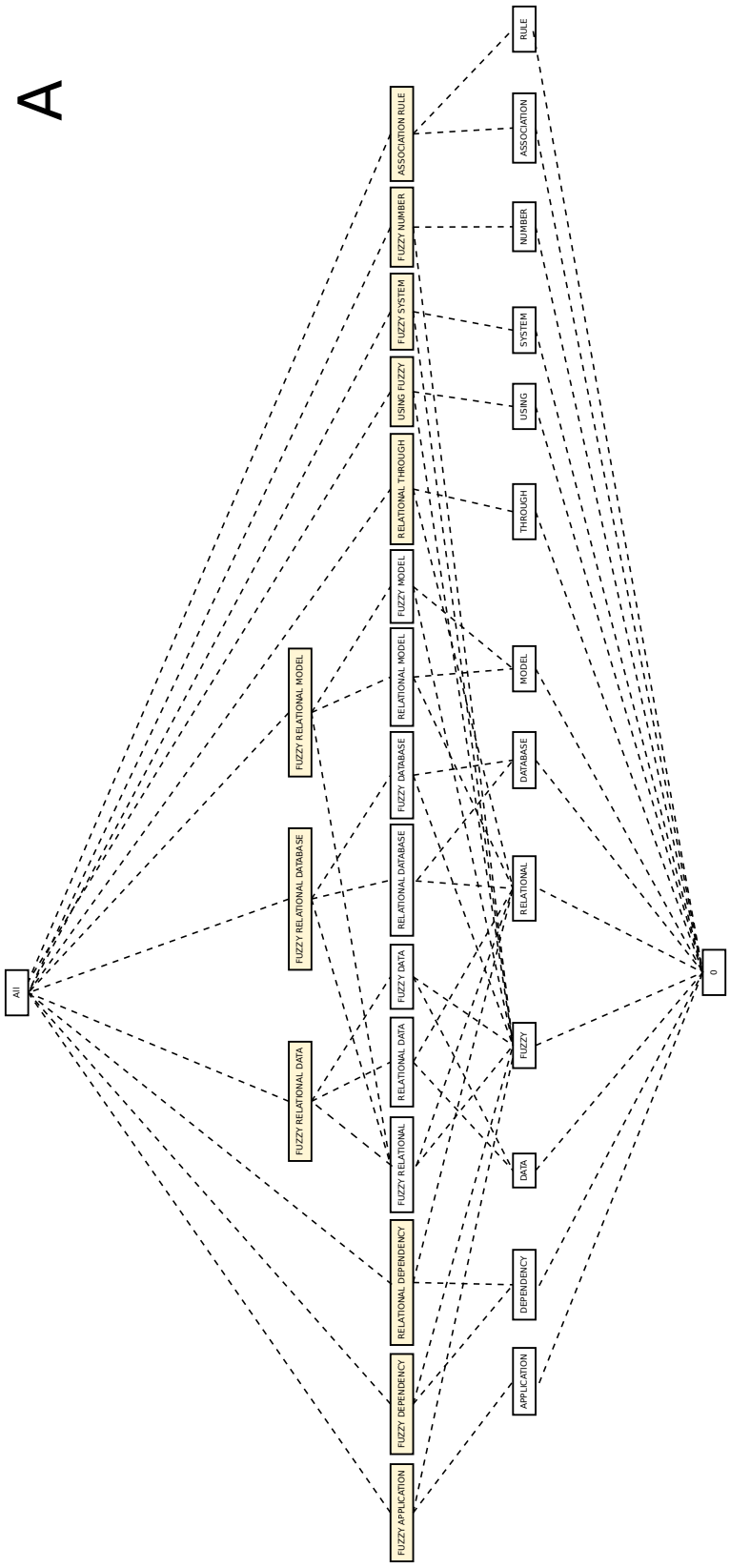


Figure 3.6: Summarized lattice defined by the AP-Structure

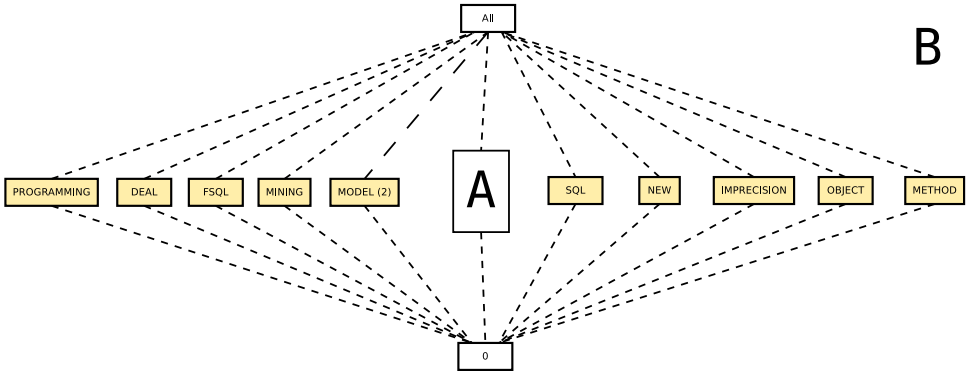


Figure 3.7: Summarized lattice defined by the AP-Structure

showing the most interesting information.

Figure 3.8 depicts the lattice formed by maximal itemsets `FUZZY RELATIONAL DATA`, `FUZZY RELATIONAL DATABASE`, `FUZZY RELATIONAL MODEL`, `SQL` and `FSQL`. It is important to remind that the ordering between terms in the itemset is not taken into account by the generation procedure. Figure 3.8 shows the itemsets ordered to ease comprehension, but the ordering issue is taken into account in a later section. Although not depicted in the figure, it is important to point out that the lattice has a lower level containing the empty set, and an upper level which is the union of all maximal itemsets.

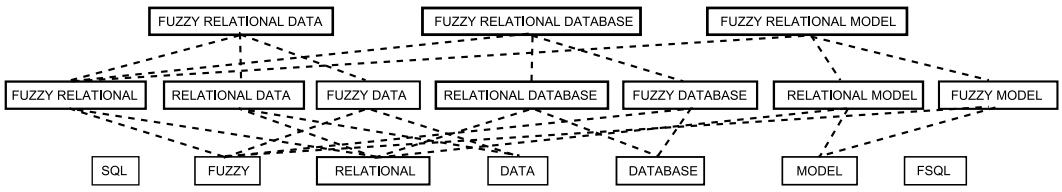


Figure 3.8: Fragment of the lattice defined by the AP-Structure

The lattice obtained is the structure that will be used as a basis for the ontology creation. To ease the example comprehension we will use the reduced version of the lattice shown in Figure 3.8.

3.2.4 AP-Structure Semantic Extension using WordNet

In this section we present a different approach to build an ontology from the lattice defined by an AP-Structure using the lexical database WordNet.

As in the previous case, we start with the whole lattice obtained from the AP-Structure. A term with cardinality value of one, which is located in the first level, corresponds with a possible concept in the ontology, because after semantic preprocessing the term is the canonical representative of a *synset* defining a concept. So, we start to build the ontology generating a class from each term in the first level of the domain AP-Structure. Once

classes are defined, the ontology is enriched with relations and new classes using WordNet semantic relations.

First, a taxonomy like hierarchical structure is built from the classes identified. WordNet relations of hyponym/hypernym between *synsets* are searched for the classes defined. For each class (represented as a *synset* in WordNet) all hypernyms are obtained. Hypernyms common to two or more classes generate a new class in the ontology. This class is defined as the parent class of those classes sharing it as hypernym. Classes with no superclass get `owl:Thing` as its parent class. This process configures a preliminary basic ontology that can be enriched with semantic relations extracted from WordNet.

Table 3.2 shows the equivalence between some of the OWL ontologies properties and the semantic relations available in WordNet.

Table 3.2: Matching between WordNet and OWL

OWL	**WordNet**
owl:Class	Synset
rdfs:subClassOf	Hyponyms/Hypernyms
owl:inverseOf	Antonym
owl:Thing	Superclass for orphan concepts

The ontology obtained using common hypernyms is not descriptive enough, because due to the linguistic nature of WordNet, the common hypernyms are abstract general terms. So we include not only the common hypernyms, but also the path to the classes. The classes are considered as subclasses of the last class in the path obtained. Although hypernym and hyponym relations are lexical relations and can not be considered equivalent to relations of inclusion between concepts, using them to form a hierarchical structure is very useful.

This approach has important disadvantages. The use of WordNet makes disambiguation a key point in the generation of correct structures. A word disambiguated incorrectly can introduce non relevant terminology and relations in the ontology. Besides, a very important drawback it is the lack of technical vocabulary on WordNet. A lot of terminology is missing and sometimes the ontologies generated are not relevant enough. Individual terms not found in WordNet are included as subclasses of `owl:Thing`, compound terms are searched in WordNet and are ignored if not found. Maximal itemsets guarantee that all their terms and their combinations are frequent. So, if a compound term is not found, we can search for its components i.e. if `FUZZY RELATIONAL MODEL` is not found, we search for `FUZZY RELATIONAL`, `FUZZY MODEL`, `RELATIONAL MODEL`, `FUZZY`, `RELATIONAL` and `MODEL`.

The ontology obtained using this approach is very large and complex. Due to space limitations, Fig. 3.9 only shows the path containing terms related to the itemsets shown in future examples.

As can be seen, most technical compounds are not found, `FUZZY RELATIONAL DATA`, `FUZZY RELATIONAL DATABASE` and `FUZZY RELATIONAL MODEL` do not appear in the resulting ontology. The exception is `RELATIONAL DATABASE`. Specific vocabulary is lost; when an

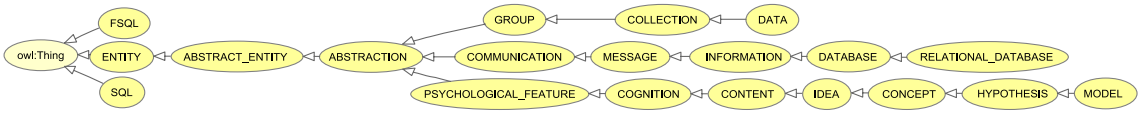


Figure 3.9: Fragment of the ontology obtained using WordNet

individual term is not found in WordNet, the concept created is attached to the root node. This means that no additional semantics is found. Each class is annotated with synonyms extracted from WordNet to improve syntactic term driven search. This approach has the advantage of providing new relevant terminology to the context taking into account semantics, but fails when the vocabulary is very specific or the disambiguation process it is not appropriate.

Combining WordNet extension with the basic structure obtained from the AP-Structure lattice it is possible to include unknown technical terminology and the addition of relevant semantic meaning to the concepts and its relations.

A lattice structure can be seen as a basic inclusion ontology. The lattice provides some basic hierarchical structure that needs to be filtered in order to be of use. The following sections describe different methods to obtain an ontology from the lattice defined by the AP-Structure obtained.

AP-Structure Semantic Extension using Syntactic Rules

Our first approach considers the lattice as a inclusion ontology. The problem is that it contains information not relevant for the ontology construction and furthermore the order of terms inside itemsets is not fixed. As explained before, when the AP-Structure is built, relative order between terms conforming an itemset is ignored. So, the only information we can rely is that they are frequent and usually appear in the same context.

We can provide a certain order to terms in an itemset following a set of basic naive syntactic rules. These ordered itemsets can be seen as candidates to be a concept in the ontology.

The set of rules applied are the following:

- **Level 1 :** Individual terms are considered candidates to be a concept in the ontology if they are nouns [N] or verbs [V].

- **Level 2 :** The itemsets composed by two terms, are considered candidates to be a concept in the ontology if they are nouns preceded by an adjective [AN] or compound nouns [NN].

- **Level N :** Candidate itemsets are those which are composed by a valid combination from the previous level and are preceded by an adjective or followed by a noun. As an example, valid combinations for level 3 are [AAN], [ANN] and [NNN].

When a node in the lattice does not follows these rules, the node is removed. Once all nodes are tested, the remaining nodes are part of a connected structure. The node in Level 0, the empty set node is removed.

In order to transform the resulting structure to an ontology, each term in level 1 is made a concept subclass of the `owl:Thing` class. For each level, nodes are included as concepts in the ontology and relations between itemsets are treated as hierarchical ISA relations.

Following this approach we obtain the fragment of ontology shown in Fig. 3.10.

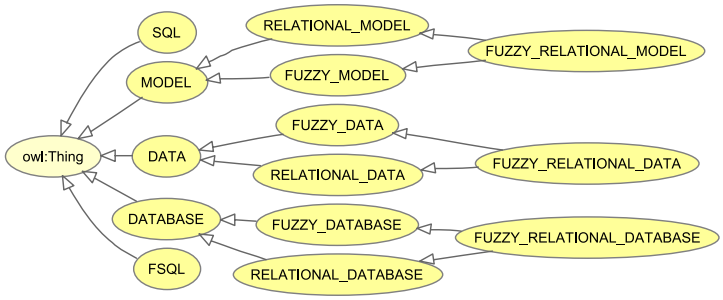


Figure 3.10: Fragment of the ontology defined by the AP-Structure

As can be observed from the figure, the ontology presents a reasonable structure, however from the point of view of querying this ontology does not add any value to the process because this structure is equivalent to a search conducted by keywords. Nevertheless the ontology can be of great use for a user who does not know anything about the context of the text field to be searched.

Since words in the concepts of the ontology are POS tagged and are disambiguated, we include as annotation properties of a class its synonym terms in order to enrich syntactic search. This enhances the queries with new syntactic terms by allowing to search for a particular concept using the different syntactic forms used to refer to it.

This approach only takes into account mathematical properties and procedures, and basic naive syntactic rules, in order to obtain some semantics from the structure we propose another approach based on the use of automated lexical dictionaries.

Combined AP-Structure Semantic Extension

This combined approach tries to compensate the virtues and drawbacks of previous approaches. The idea is to be able to deal with unknown terminology while giving proper semantics to known terminology. The problem concerning disambiguation is still there and the only solution is to look for the appropriate disambiguation technique.

This approach starts as the naive approach, creating the basic ontology, taking into account syntactic rules. Once the basic ontology is built, WordNet is used to search for common hypernyms and its paths. The result is an extended ontology containing unknown terminology correctly placed under the known terms.

Figure 3.11 shows the same fragment of the ontology as Fig. 3.9 but obtained using the combined approach.

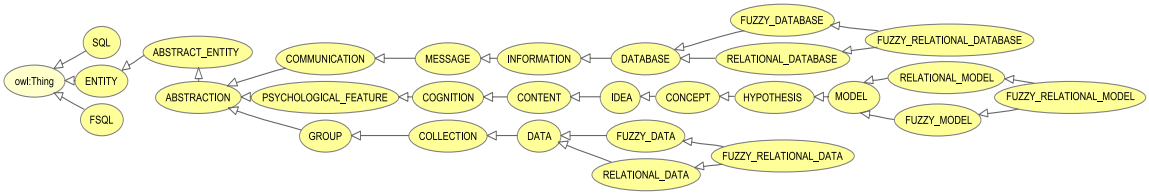


Figure 3.11: Fragment of the ontology obtained using the combined approach

As can be seen, unknown terminology is preserved and basic terms are extended using WordNet.

The automatic extraction of ontologies from textual attributes allows the description of textual attributes whose content is unknown to the user. The ontologies generated in the proposal contain a list of search terms for each concept defined in the ontology. These terms are extracted from WordNet and are synonyms of the concept modeled in the ontology. The search terms are encoded as annotation properties, and can be used to build queries over the processed text attribute.

This approach presents a simple ontology that can be used to enhance querying providing additional terms and allows to provide certain semantics to the domain comprised by the textual attributes processed. Similarly the ontology graphical representation can be used as a guide to perform queries, because each concept has several search terms associated.

Although the use of WordNet allows to extend the original lattice with some semantics, the bias of WordNet taxonomy to linguistic representation tends to include general concepts not relevant to the problem domain. In order to explore other knowledge sources, next section analyzes Wikipedia as a tool for semantic extension.

3.3 Semantic Representation of Non-Structured Texts using Wikipedia

The extension of the AP-Structure using WordNet yields poor results due to the use of very general and abstract linguistic oriented terms. The structure of WordNet is designed based on linguistic principles. This structure it is not the most appropriate for our purposes, so it is convenient to choose another source of knowledge whose structure suits our needs.

When selecting a knowledge source to extend the AP-Structure we must take into account that the source is structured, well defined, and designed from a practical point of view. Lots of ontologies present these requirements but they lack of a proper maintenance and frequent updates.

Following the methodology previously presented, it is possible to use other knowledge sources to generate the ontology. In this case we use Wikipedia, more precisely its category graph, to obtain information.

The steps performed for the generation of an ontology from non-structured text using Wikipedia, comprises a series of steps. The first step is to obtain the AP-Structure from the texts. The process is equivalent to the one previously shown for WordNet.

Let us summarize again the process:

- Determine the text attribute in the database to process.
- Preprocess data using the appropriate filters.
- Semantic preprocessing. This stage requires:
  - POS Tagging.
  - Word sense disambiguation.
  - Substitution of the original words for the form selected as canonical representative of the *synset* to which the word belongs.
- Once the pre-processing stage is finished, create the domain AP-Structure. This domain AP-Structure represent a summary of the original texts.

From the AP-Structure obtained we obtain the ontology. The whole process adapting the general methodology to the case of using Wikipedia as a resource is shown in 3.12.

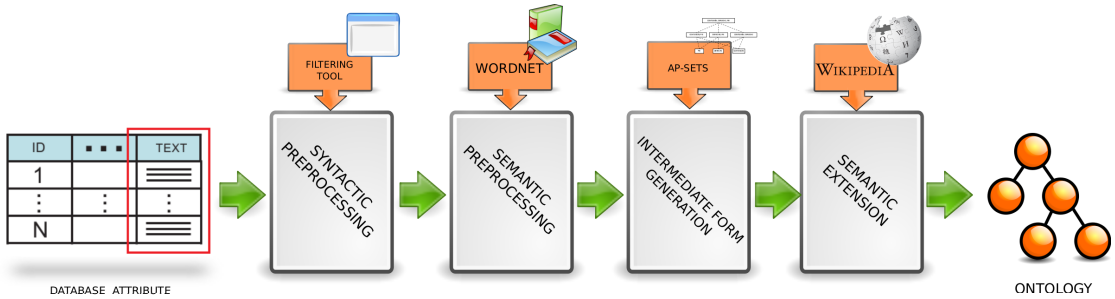


Figure 3.12: Wikipedia based semantic representation of non-structured texts

In order to select the proper categories from Wikipedia category graph we present several methods for category selection. Before presenting further details about these methods, let us see a brief description of the Wikipedia category graph.

3.3.1 Wikipedia Category Graph

Wikipedia is an online collaborative encyclopedia where users can generate and edit contents. Each page of Wikipedia is associated with one or more categories. In regard to categorization Wikipedia states: *"Categories are pages that are used to group other pages on similar subjects together. Wikipedia's category system can be thought of as consisting of overlapping trees. Any category may branch into subcategories, and it is possible for a category to be a subcategory of more than one parent. (A is said to be a parent category of B when B is a subcategory of A). Mathematically speaking, this means that the system*

approximates a directed acyclic graph. There is one top-level category, Category:Contents. All other categories are found below this. Hence every category apart from this top one must be a subcategory of at least one other category.".

Categories are connected to each other by relations of hyponymy/hypernymy. All relations are defined by Wikipedia editing users. The Wikipedia graph has a similar tree structure as that of WordNet, and can be seen as a thesaurus combining collaborative tagging and hierarchical indexation. The main drawback of Wikipedia category graph is the existence of cycles. Some categories can be defined as super-categories and subcategories of a given category at the same time. This kind of collaborative structures is studied in [Mehler, 2009] where they are described as *Social Ontology Graphs (SOG)*, where the term social refers to the way in which the ontology is built.

Wikipedia category graph is defined as:

Definition 3. *Wikipedia category graph G_W is defined as a directed graph $G_W = (V, A, r)$, where V is a set of vertex, each vertex maps to a category in Wikipedia, A is a family or arcs defining hierarchical relation between categories, and r represents the main Wikipedia category, the root node of G_W. This graph has two functions defined $init : A \to V$ and $ter : A \to V$ assigning each arc $a \in A$ to a initial vertex $init(a)$ and a final vertex $ter(a)$. This represents an arc a directed from vertex $init(a)$ to vertex $ter(a)$.*

A Wikipedia page is defined as follows:

Definition 4. *A Wikipedia page is defined as $p = (t, c, L, LC)$, where t is the title, and c is the page content, $L = \{l_1, l_2, \ldots, l_n\}$ is the set of links referencing other pages and $LC = \{lc_1, lc_2, \ldots, lc_n\}$ is the set of link referencing the categories on which the page is classified.*

Figure 3.13 shows a Wikipedia page with all the components, title, content, links to pages and links to categories.

Each category contain information about the pages on it, then a category can be defined by its pages. Each page can be seen as a label describing a topic related to the category.

Definition 5. *A Wikipedia category $C = (P, LC)$ is defined as a set of Wikipedia articles or pages $P = \{p_1, p_2, \ldots, p_n\}$ and a set of links to its parent categories $LC = \{lc_1, lc_2, \ldots, lc_n\}$.*

The intersection of the pages in two different categories can be not empty, as a page can be categorized in more than one category.

Figure 3.14 shows a simple depiction of a category structure, containing links to its parent categories and a set of pages belonging to the category.

The Wikipedia category graph is a complex structure, its huge size and the high level of interconnection between its nodes makes its automatic processing a non-trivial task. If we get all the connections for a given random term, selecting all the categories to which it is connected and the ones connected to them recursively, the probability of ending up with a graph with a similar size to the whole Wikipedia graph is very likely.

We intend to create an ontology where AP-Structure terms are contextualized. This ontology is going to be used by human users, so the size must be relatively small in order

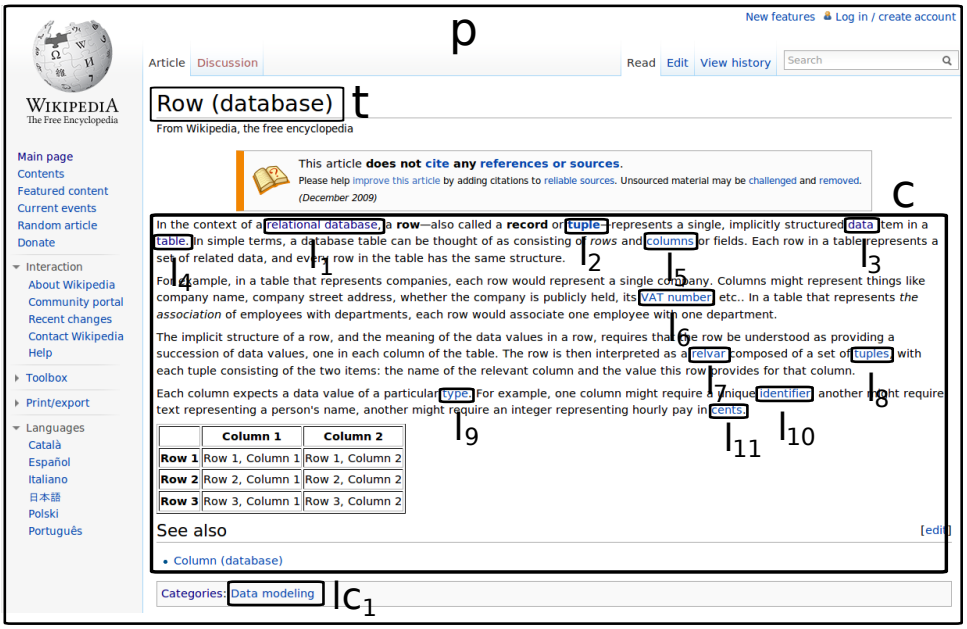


Figure 3.13: Components of a Wikipedia page

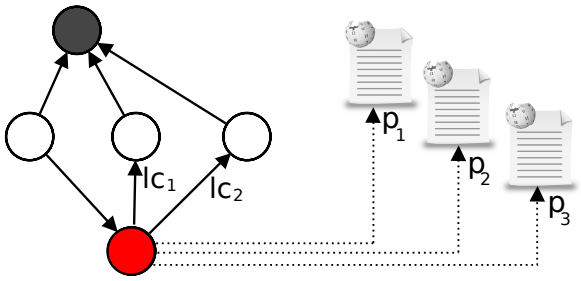


Figure 3.14: Components of Wikipedia Category

to avoid overwhelming the user with too much information. To do this, we propose to select a sub-graph of the whole Wikipedia graph which properly defines a term or a set of terms taken from an AP-Structure. The selection of this sub-graph will be performed by using semantic distance, semantic similarity and information measures.

In order to select the appropriate Wikipedia categories we need to compare them. Our work is focused on exploiting information using three different approaches:

- Graph exploration: Starting from a initial category a path to the root is travelled. Categories are selected along this walk to the root. These methods are depicted in detail in next sections.
- Semantic structure: Using WordNet semantic similarity measures adapted to Wikipedia it is possible to obtain similarity values between categories attending to topological criteria of the underlying structure. The similarity value obtained allows to discard categories loosely related to the initial ones. Getting only the related categories allows to create a structure coherent with the initial categories obtained from the terms in the AP-Structure.
- Information retrieval: Each of the Wikipedia categories has a set of pages. This pages contain a title and text content. Using information retrieval techniques it is possible to determine similarity between two categories according to their content. This comparison can be conducted at different levels.
    1. At page level: Each page is processed as an unique entity, hence the category can be seen as a set of pages. In order to compare the pages between categories we take the whole title as reference. A category then is reduced to a set of page titles.
    2. At page title level: In this case page titles are processed and terms are extracted and represented as a bag of words which defines the category. Using these bag of words categories can be compared.
    3. At page content level: This case takes more processing as all text in pages is expressed as a bag of words associated to the category. Comparing two categories can be reduced to the comparison of their respective bag of words using some of the metrics available.

All methods proposed use the Wikipedia category graph. In [Torsten Zesch, 2007] the Wikipedia Category graph is extensively tested and it is proved that it is a valuable tool as other electronic dictionaries. The work also presents some statistics comparing size and coverage of Wikipedia to that of WordNet. The version used was the Wikipedia dump from May the 15th 2006, in its German version. The statistics where obtained from the biggest connected component of the graph, which covered around 99.8% of the total size of the original graph. This version contained 27865 nodes, with an average of 3.54 arcs connected to a node, a diameter of 17 (the diameter is computed as the length of the biggest of the minimum paths), and the average path length between two random nodes is of 7.18 arcs. These same metrics yield values of obtain a valued of 122005 nodes, a diameter of 27 and an average path length between two nodes of 10.56 for WordNet.

3.3.2 Semantic Extension of AP-Sets using Wikipedia

Once obtained the AP-Structure, in the same way as we did in the WordNet case, we can start to extend the structure with new knowledge extracted from Wikipedia.

In order to extend the AP-Structure we must map each term in the basic inclusion hierarchy to a Wikipedia category. First, we try to map the term to a category using its title. If it is not possible to find a proper category we try to find a page for the term. For each term, we look for a page with a title similar to the term, once the page is retrieved we get the categories associated to the page and select the most appropriate. If no page is found, only the information available in the basic inclusion ontology is added to the result. When all categories are identified, a path to the root from each category is traversed and the appropriate categories are selected.

Figure 3.15 depicts the mapping between the AP-Sets containing itemsets in the AP-Structure and the nodes in the Wikipedia category graph. The complete process implies the generation of the basic inclusion ontology according to the syntactic rules presented in section 3.2.4 an then associate each of the ontology concepts to a Wikipedia category. This intermediate step is performed in order to force some order in the items of an AP-Set, but at high level the process only implies a mapping between the AP-Sets and the categories of the graph.

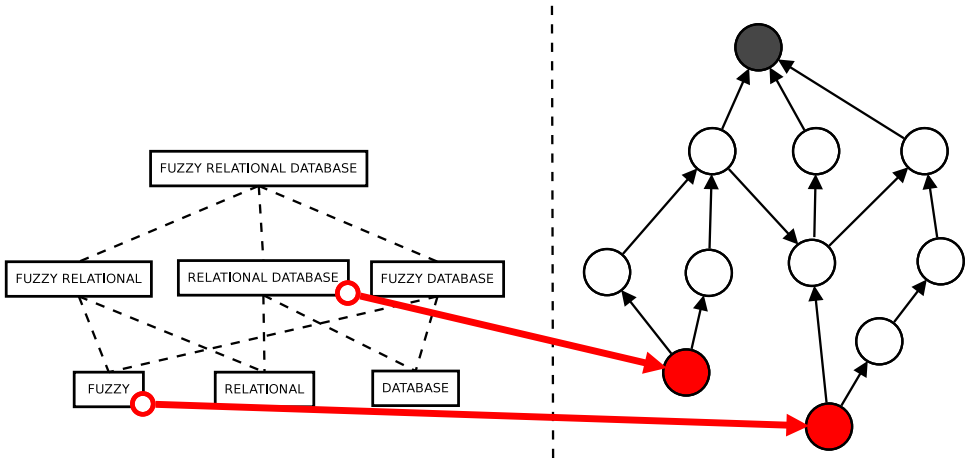


Figure 3.15: AP-Structure to Wikipedia Category Graph mapping

As the connectivity in Wikipedia category graph is high, it is necessary to define appropriate criteria to select paths between pairs of categories. These methods are presented in following sections.

All the tests have been conducted over a dump of the English Wikipedia from August the 17th 2010, stored in a MySQL database.

3.3.3 Graph Exploration Methods

This section presents different approaches for traversing a graph from a starting point towards the root. As each term in the AP-Structure represents a category and hence a category, we will look for those nodes providing additional information to the categories selected at the beginning.

Basic Exploration Method

This method does not use any additional processing. Starting from the selected categories, a path to the root is traversed. To avoid cycles a list of visited nodes is maintained. Once a node is processed, it is not processed again.

This method obtains a hierarchical structure containing all categories for the terms in the AP-Structure. Although this ontology is valid, its size makes it not very useful.

Figure 3.16 depicts an example of graph exploration towards the root. Nodes in color are start nodes, and those with thick lines are nodes visited while traversing the graph. The root is present in solid black. As this example shows, when the graph is highly connected almost all the nodes are visited in the way to the root.

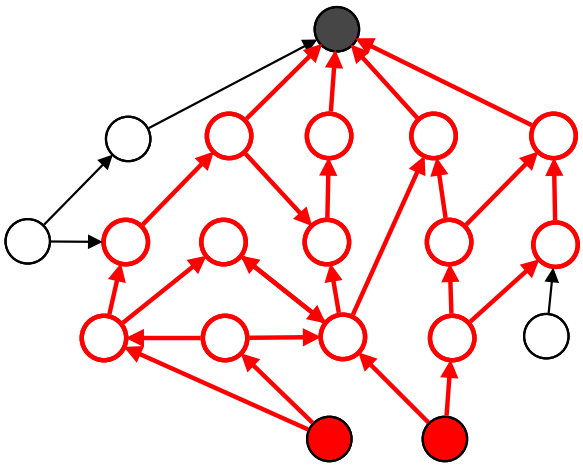


Figure 3.16: Basic Exploration Method

The formal description of this process is presented in Algorithm 5.

In order to ease comprehension of the examples performed, we just select two elements from the lattice shown in Figure 3.5. We select the first level element *SQL* and second level element *Association Rule*. This selection is performed to show the results obtained in a simple way. Later, these short examples can be generalized to more complex and general ones. The term *SQL* y mapped to the category *SQL* in Wikipedia, and the term *Association Rule* is mapped to the category *Data Mining*. From these categories a path to the root is traversed adding all the categories in the path to the final ontology. The ontology obtained has the characteristics depicted in Table 3.3.

As can be observed, the number of categories selected in the path to the root is very

Algorithm 5 Basic Exploration Algorithm

Require: Graph $G_W = (V, A, r)$ and a set of start nodes $Vc = \{v_1, \ldots v_n\}$
Ensure: The set of nodes Vs and the set of arcs As

$Vs_0 \leftarrow \emptyset, As_0 \leftarrow \emptyset$
$Vs_1 \leftarrow v_l, As_1 \leftarrow \emptyset$
$k \leftarrow 2$
for all $\{v_i \in Vs_{k-1} \wedge v_i \notin \bigcup_{j=0}^{k-2} Vs_j\}$ **do**
  if $\{a \in A, v \in V : (init(a) = v_i) \wedge (ter(a) = v)\}$ **then**
    $As_k \leftarrow a, Vs_k \leftarrow v$
  end if
  $Vs_k \leftarrow [Vs_k \backslash Vs_{k-1}]$
  $k++$
end for
$As \leftarrow \bigcup_{i=1}^{k-1} As_i, Vs \leftarrow \bigcup_{i=1}^{k-1} Vs_i$
return As, Vs

Table 3.3: Number of ontology classes for the basic method

Original terms	*SQL*	*Association Rule*
Base Category	SQL	Data Mining
Number of Nodes/Categories	2786	2784
Average number of Parent Nodes	8	8
Maximum number of Parent Nodes	14	14
Average number of Sibling Nodes	15	15
Maximum number of Sibling Nodes	42	42
Maximum Depth	56	53
Minimum Depth	5	3

high. This happens because the areas close to the root in the Wikipedia category graph are highly connected.

As this method does not yields appropriate results, we propose other methods to select the appropriate categories.

Summarized Basic Method

This method starts as the previous one, from a set of selected categories. This time, instead of expanding all nodes found in the path to root, we search for common parents to the start nodes, or as we call them Base Categories. In [Wu and Palmer, 1994] the concept of *LCS* is introduced. The LCS of a set of nodes is the *least common subsumer* of those nodes, that is the nearest parent of those parents common to all nodes in the set.

This way we can relate the different terms and discard all non relevant nodes. The process starts trying to find a LCS for the base categories, subsequent iterations try to find the LCS for those nodes that still do not have one and for the LCSs found in the previous iterations. The process ends when no LCS are found, or the LCS for the current set is the root node.

Figure 3.17 shows the method applied over the example graph previously used. Start nodes are drawn with solid color, thick lines depict the path to the LCS, and dashed lines model new arcs not present in the original graph.

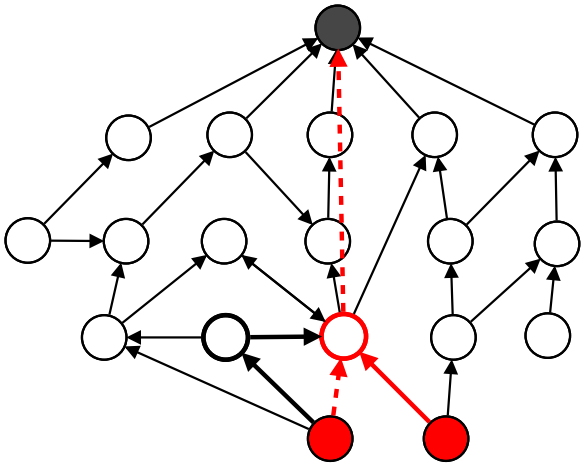


Figure 3.17: Summarized Basic Method

The formal description is presented in Algorithm 6. It is important to take into account that if the LCS is the root node (r), that means that there is no LCS between such nodes.

This method ensures the reduction in the size of the final ontology. One drawback is the loss of all the terminology provided by the intermediate classes. Other drawbacks are particular cases where the LCS of a set of nodes is one of the nodes in the set, this causes that no new terminology is added to the results. When new categories are added, sometimes they represent general and not much relevant concepts that do not add meaningful

Algorithm 6 Summarized Basic Exploration Algorithm

Require: The graph $G_W = (V, A, r)$ and a set of start nodes $Vc = \{v_1, \dots v_n\}$
Ensure: The set of nodes Vs and the set of arcs As

$Vs_1 \leftarrow Vc, As_1 \leftarrow \emptyset$
$k \leftarrow 1$
while $Vs_k \neq \emptyset$ **do**
  for all $\{v_i \in Vs_k\}$ **do**
    for all $\{v_j \in Vs_k : j > i\}$ **do**
      $l \leftarrow lcs(v_i, v_j)$
      if $l \neq \emptyset$ **then**
        $Vs_{k+1} \leftarrow l$
        $As_{k+1} \leftarrow b : init(b) = v_i \wedge ter(b) = l$
        $As_{k+1} \leftarrow b : init(b) = v_j \wedge ter(b) = l$
      end if
    end for
  end for
  $Vp = \{b \in As_{k+1}, v \in Vs_k, w \in Vs_{k+1} : (init(b) = v) \wedge (ter(b) = w)\}$
  $Vs_k \leftarrow [Vs_k \cup Vs_{k+1}] \backslash Vp$
  $k++$
end while
for all $\{v_i \in Vs_{k-1}\}$ **do**
  $As_{k-1} \leftarrow b : init(b) = v_i \wedge ter(b) = r$
end for
$As \leftarrow \bigcup_{i=1}^{k-1} As_i, Vs \leftarrow \bigcup_{i=1}^{k-1} Vs_i$
return As, Vs

information. Some of the categories close to the root are of this type, too general and almost meaningless.

Figure 3.18 presents the ontology obtained by applying the method to the terms *SQL* and *Association Rule*. As can be seen, new common classes provide more semantics to the ontology, but it is not enough to consider this the most appropriate method to generate relevant ontologies from the terms.

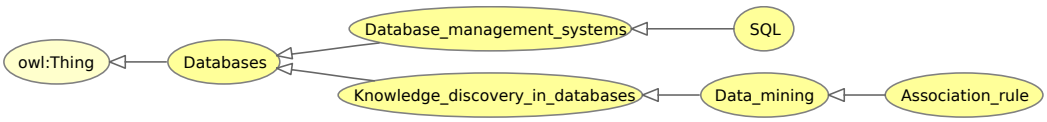


Figure 3.18: Least Common Subsumer

Path Length Methods

As the size of the structure of Wikipedia Category Graph is a big challenge, we can work around the problem establishing limits to the graph exploration. This imposed limitation is modeled as an additional parameter n, an integer value representing the maximum length to explore. This way, general nodes close to the root can be avoided.

This approximation has several advantages, but their drawbacks are important too. The main advantage is the great number of category nodes left aside, making the ontology reader friendly. Moreover, the parameter n determines a bounded size of depth for the ontology. These methods are based on topological measures; although these measures can determine the relatedness of concepts, it is necessary to use them in conjunction with semantic measures in order to obtain better results.

The most important drawback is the existence of a LCS out of length n, as a LCS provide contextual meaning to their children. This can be solved using two variants, a path length method or a summarized path length method.

Path Length Basic Method

The method starts with the base categories and explores the graph in a distance equal or less to the parameter n. All categories whose distance to a base category is less than n are ignored.

Figure 3.19 shows the selected nodes in the example graph for a path length of $n = 2$. Nodes in solid color are exploration start nodes, arcs with thick lines are those selected in the path. Arcs in dashed lines depict new arcs created to connect with the root node. As can be seen in the figure, there are nodes whose distance to one base node is greater than n, this is due to the existence of an alternate path at the right distance.

The formal description of the method is presented in Algorithm 7.

It is important to select an appropriate value for the parameter n. Depending on the depth of the base category in the Wikipedia category graph, a high value for n could lead

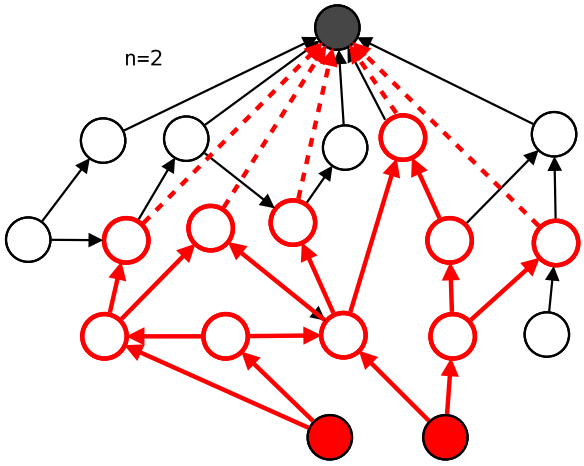


Figure 3.19: Path Length Basic Exploration Method

Algorithm 7 Path Length Basic Algorithm

Require: The graph $G_W = (V, A, r)$, the start nodes $Vc = \{v_1, \dots v_n\}$ and length n
Ensure: The set of nodes Vs and the set of arcs As

$Vs_0 \leftarrow \emptyset, As_0 \leftarrow \emptyset$
$Vs_1 \leftarrow Vc, As_1 \leftarrow \emptyset$
for $k \leftarrow 2$ **to** $n+1$ **do**
  for all $\{v_i \in Vs_{k-1} \wedge v_i \notin \bigcup_{j=0}^{k-2} Vs_j\}$ **do**
    if $\{a \in A, v \in V : init(a) = v_i \wedge ter(a) = v\}$ **then**
      $As_k \leftarrow a, Vs_k \leftarrow v$
    end if
    $Vs_k \leftarrow [Vs_k \backslash Vs_{k-1}]$
  end for
end for
$As \leftarrow \bigcup_{i=1}^{n+1} As_i, Vs \leftarrow \bigcup_{i=1}^{n+1} Vs_i$
return As, Vs

to the highly connected area of the graph, and therefore introducing a high number of new categories.

Let us analyze the behaviour of the method when changing the n parameter for the terms *SQL* and *Association Rule*. Figure 3.20 shows the values of size for the final ontology as opposed to the parameter n. As can be observed, for small values of n the growth of the ontology increases in linear form, and suddenly starts to grow quickly. This point of sudden growth coincides with the point where the exploration enters the highly connected zone of the Wikipedia category graph. Thus, an appropriate value for n cannot be predicted beforehand, as it depends of each term and the position of its corresponding category in the graph.

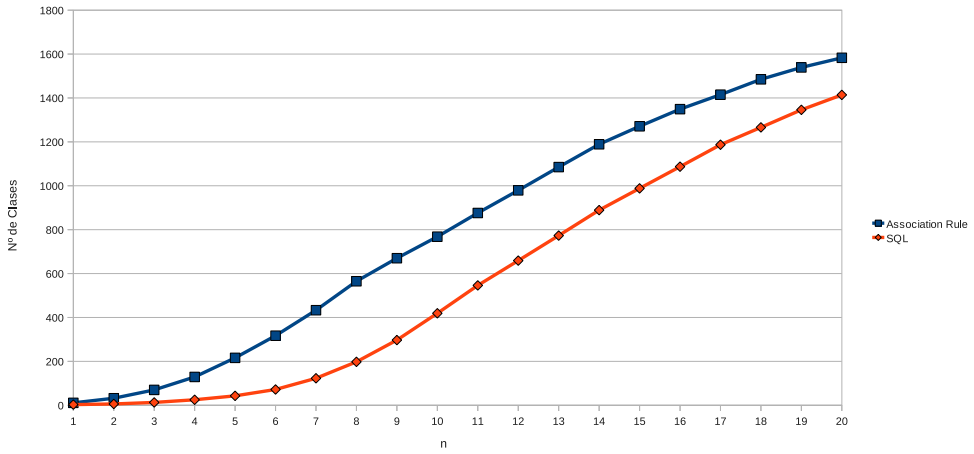


Figure 3.20: Number of Nodes for different values of n

Summarized Path Length Basic Method

This method relies in the summarized basic method, the intermediate node categories between a category and its LCS are expanded according to a parameter n and a selection policy. The n parameter determines the maximum amount of intermediate categories to select, and the selection policy determines which categories to select. These parameters will vary substantially the organization of the information in the ontology.

We propose four selection policies:

1. Selection of the closest n super-categories to the base category: This way the context closest to the base category is better defined. Parent categories are used to contextualize relations with the base category.

2. Selection of the closest n subcategories to the LCS: This way the context of the common category is better specified. The surroundings of the base category are left unspecified.

3. Selection of the closest n categories to base category and the common category: This variant allows to contextualize both categories but ignores intermediate categories in

the path between the base category an the LCS. In case that parameter n is odd, the last category is selected from the context of the base category.

4. Selection of n categories in the path between the base category and the LCS : This variant characterizes the whole path form a base category to the common category.

It is also possible to expand all the paths, not only those leading to a base category. This would expand all paths between related parent categories.

Figure 3.21 shows the result of applying the method to an example graph using a length o $n = 2$ and the different selection policies. Nodes in solid color are start nodes, LCS node is highlighted with a thick line contour. Thick lines represent existing arcs that have been selected, and dashed lines represent newly created arcs.

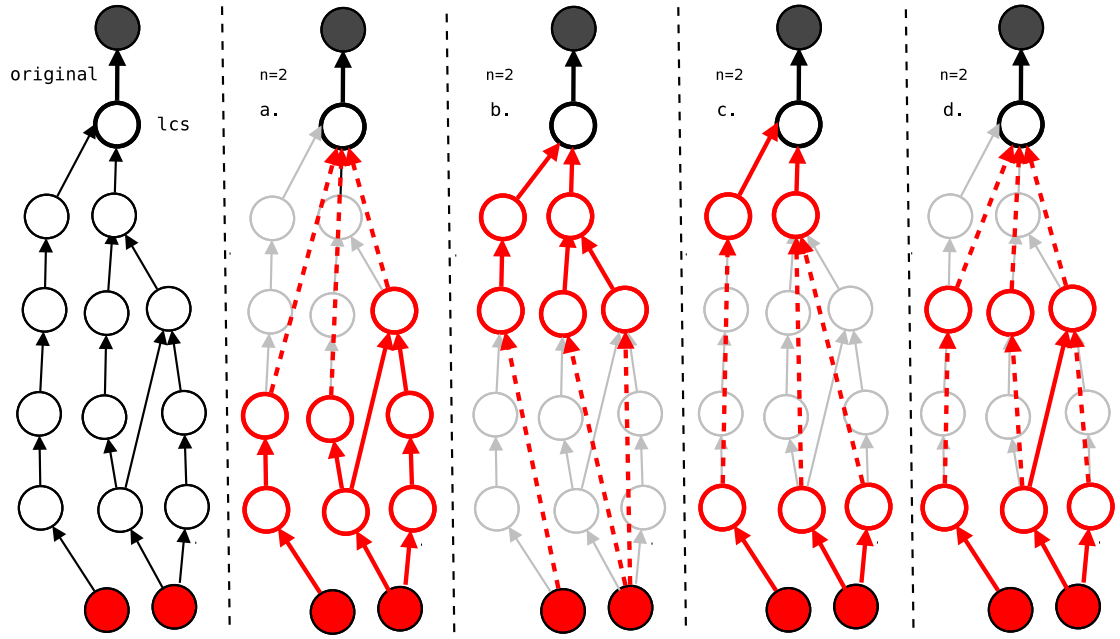


Figure 3.21: Path Length Summarized Basic method with $n = 2$ and different policies

It is important to point out that the variants are applied over each one of the possible paths between the start node and the LCS, as each of these paths contributes to that particular node.

Figure 3.22 displays the ontology obtained for the terms *SQL* and *Application* using the current method with parameter $n = 2$ and different policies.

Although all methods yield appropriate results, the first selection policy seems to provide more coherent and contextualized results for the terms used. This proves that increasing the generalization it is not reflected as an increase of relevant information in the ontology. It seems that the exploration of categories near to the base category provides more information than exploring the categories near to the LCS.

This method stops, like the previous one, when no more LCSs can be computed. This leaves most of the graph unexplored. We avoid entering the highly connected part of the

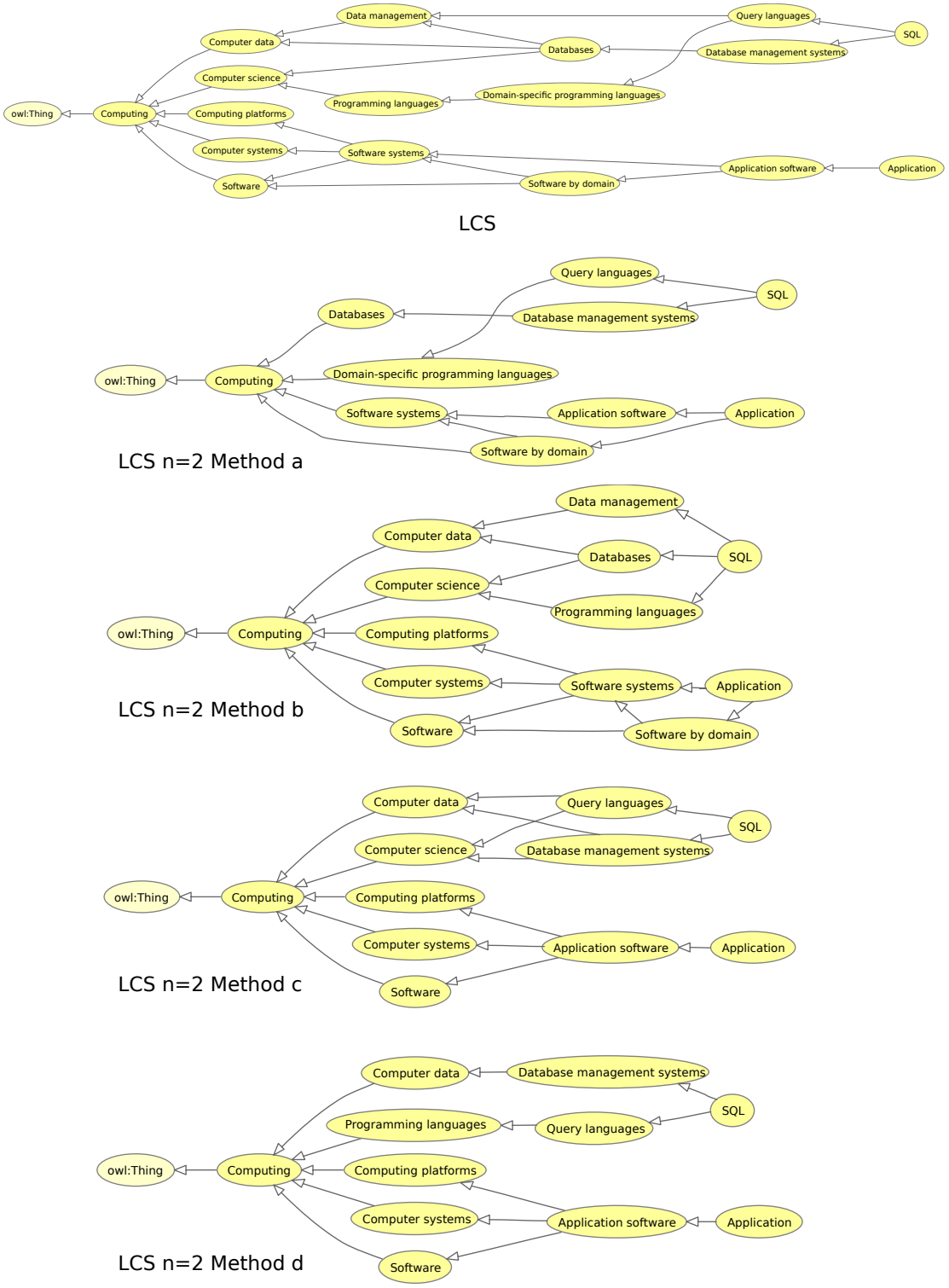


Figure 3.22: Ontologies for the terms *SQL* and *Application*

graph, but we do not get any information from it. A good possibility could be using a short path method from the LCSs to the root, in order to get more information and avoid breadth expansion.

Shortest Path Method

This method obtains the shortest path to the root from each of the base nodes, ignoring relations between the terms associated to them. This poses a drawback, common ancestor categories are ignored, but it allows to define an informative taxonomy.

Figure 3.23 shows the shortest paths selected in the example graph. Start nodes are displayed with solid color, and selected arcs and nodes are drawn in a thick line. As can be seen in the figure the node at the left has two paths to the root with the same minimum length. In this case, as we cannot decide to select one over the other, we select both.

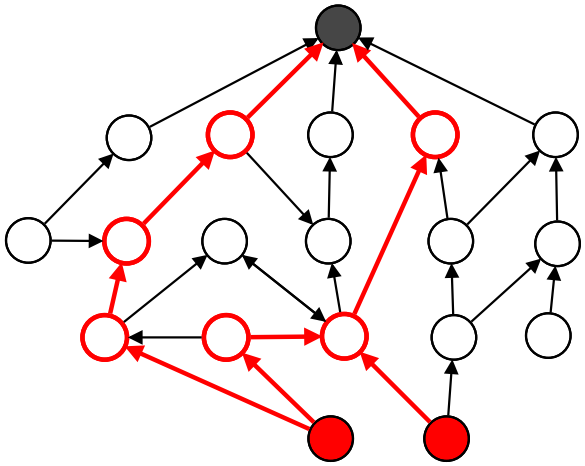


Figure 3.23: Shortest Path Method

Algorithm 8 presents a routine to obtain the shortest paths to the root from a set of start nodes. The definition for the recursive procedure to find the shortest path is depicted in Algorithm 9. The solution contains a set of nodes and arcs that are a subset of the original graph and contain all the shortest paths from the start nodes to the root.

Figure 3.24 depicts the ontology obtained with this method for the term *SQL*. There are two shortest paths to the root, both with the same length. In case that more than one shortest path exists all are selected, as the only criterion right now is the topology of the graph an thus all are equally valid.

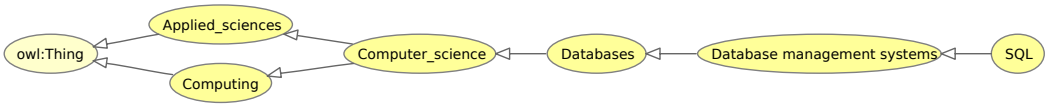


Figure 3.24: Shortest paths for term *SQL*

Algorithm 8 Shortest Path method

Require: The graph $G_W = (V, A, r)$, the start nodes $Vc = \{v_1, \dots v_n\}$
Ensure: The set of nodes Vm and the set of arcs Am containing the shortest path
  $Vs, As \leftarrow \emptyset$
  $Vp, Ap \leftarrow \emptyset$
  $Vm, Am \leftarrow \emptyset$
  for all $\{v_i \in Vc\}$ **do**
    $explorePath(r, v_i, Vp, Ap, Vs, As)$
    $Vm \leftarrow Vs$
    $Am \leftarrow As$
  end for
  return Vm, Am

Figure 3.25 shows the ontology obtained form the term *Association Rule*. As the term does not exists as a category it is represented as a subclass of the main class *Data Mining*. The path obtained is disjoint with respect to that obtained for the term *SQL*.

Figure 3.25: Shortest path for the term *Association Rule*

This method has an important drawback, the shortest paths are not necessarily the most informative ones. Larger paths could contain more relevant information than the ones selected. In the case of the term *SQL* there are other branches including relevant concepts related to programming languages and data management that are ignored.

Combined Taxonomic Method

In order to avoid the main drawbacks of previous methods, we develop a new method exploiting the virtues of previous approximations. The main problem with taxonomic methods is the exploration of the highly connected parts of the Wikipedia graph. Previously, we observed how categories adjacent to base categories contain relevant information and are solid candidates to include in the ontology. But, as we leave the base categories and traverse the graph towards the root, categories start to be more general and less related to the start ones, not to mention the risk of getting entangled in the aforementioned over-connected zone.

This method consists of the exploitation of the vicinity of base categories using path length, and then using the shortest path for all categories without parents. This way, we avoid the conflict zone using a direct path, while we allow the local exploration of zones close to the base categories.

Figure 3.26 shows the method applied on the example graph. The first step is equivalent to the path length method in Section 3.3.3. The difference resides in the fact that once explored the vicinity of the base nodes with length n, orphan nodes are not attached to the

Algorithm 9 Recursive procedure for shortest path calculation

Procedure $explorePath(r, c, Vc, Ac, Vs, As)$

Require: Root node r, current node c, current set of nodes and arcs Vc and Ac, and the set of nodes and arcs in the current shortest path Vs and As

```
Vc ← c
if {c = r} then
  if {|Vs| ≠ 0} then
    if {|Vc| < |Vs|} then
      Vs ← ∅
      As ← ∅
      Vs ← Vc
      As ← Ac
    end if
  else
    Vs ← Vc
    As ← Ac
  end if
end if
if {|Vs| ≠ 0 ∧ |Vc| >= |Vs|} then
  return
end if
Ap ← a ∈ A : init(a) = c
if {Ap = ∅} then
  return
end if
for all {a ∈ Ap} do
  n ← v ∈ V : ter(a) = v
  Ac ← a
  if {n = r} then
    return
  end if
  Vc', Ac' ← Vc, Ac
  explorePath(r, n, Vc', Ac', Vs, As)
end for
return
```

root, instead a shortest path to the root is computed as can be seen in Figure 3.19. The first step is marked in red in Figure 3.26, and the second step with shortest path search is depicted in blue. As in previous figures solid nodes are start nodes and the black node is the root of the directed graph.

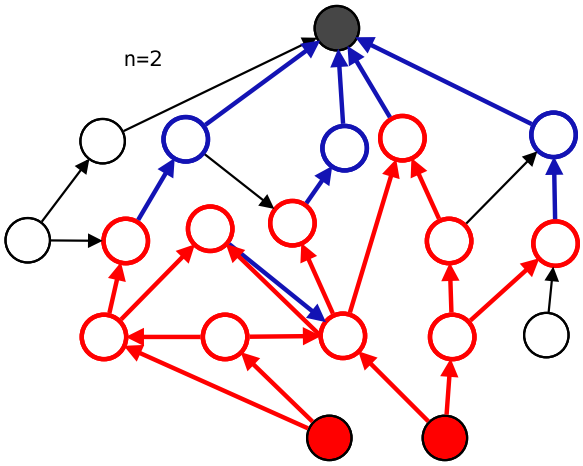


Figure 3.26: Combined Taxonomic Method

Figure 3.27 presents the ontology obtained for this method using a length of $n = 3$ for the terms *SQL* and *Association Rule*

The formal definition is presented in the Algorithm 10. The algorithm explores the graph until a length of n form the start node is reached, and then applies shortest path search as in Algorithm 9.

The main drawback of this method is the lack of a evaluation measure for each node that avoids the inclusion of non-relevant nodes.

At this point, we can foresee that selection based only in taxonomic criteria is not enough to obtain compact and meaningful ontologies. The position of the categories in the graph can alter results, so the election of the parameter for the methods is almost meaningless. We must use another criterion than mere topological position. In this context, basic exploration used in conjunction with other types of criteria could be a good option.

3.3.4 Measure based Methods

Previous section allowed us to create an ontology in a straightforward way using the information encoded in the Wikipedia category graph structure. This graph is built collaboratively by thousands of users, and its knowledge is reflected in the relations between categories. But Wikipedia category graph provides more data that can be exploited in order to determine the relation between a pair of categories.

Until now exploratory methods relied only on structural information, and so they were unable to determine the relevance of the categories selected. These methods ignored a really important source of knowledge, the pages associated to each category.

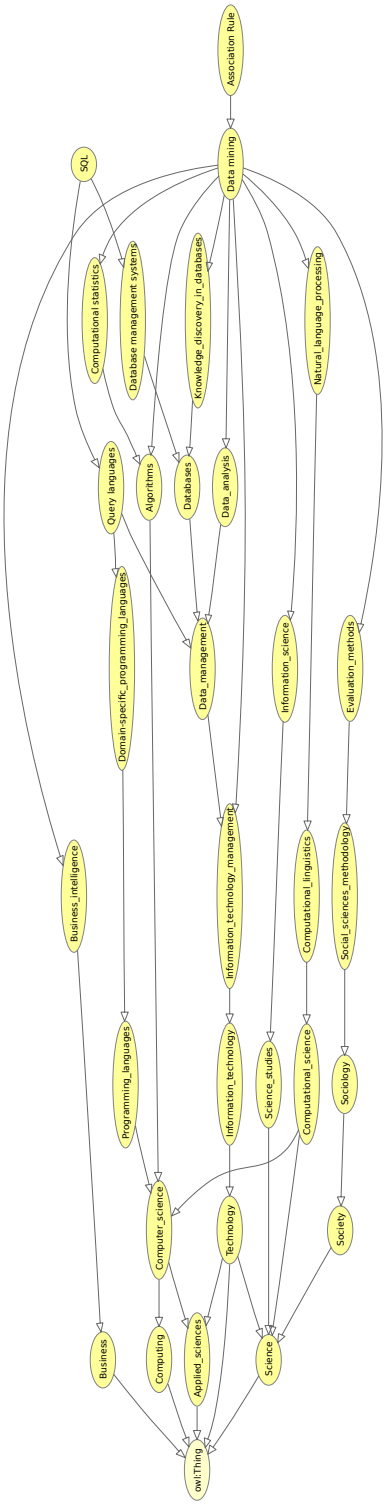


Figure 3.27: Ontology for Combined Taxonomic Method with $n = 3$

Algorithm 10 Algorithm for Combined Taxonomic Method

Require: The graph $G_W = (V, A, r)$, start nodes $Vc = \{v_1, \dots v_n\}$ and length n
Ensure: The set of nodes Vs and the set of arcs As

$Vs_0 \leftarrow \emptyset, As_0 \leftarrow \emptyset$
$Vs_1 \leftarrow Vc, As_1 = \emptyset$
for $k \leftarrow 2$ **to** $n+1$ **do**
  for all $\{v_i \in Vs_{k-1} \wedge v_i \notin \bigcup_{j=0}^{k-2} Vs_j\}$ **do**
    if $\{a \in A, v \in V : init(a) = v_i \wedge ter(a) = v\}$ **then**
      $As_k \leftarrow a, Vs_k \leftarrow v$
    end if
    $Vs_k \leftarrow [Vs_k \backslash Vs_{k-1}]$
  end for
end for
for all $\{v_i \in Vs_k\}$ **do**
  $Vp, Ap \leftarrow \emptyset$
  $Vm, Am \leftarrow \emptyset$
  $explorePath(r, v_i, Vp, Ap, Vm, Am)$
  $Vs_{k+1} \leftarrow Vm$
  $As_{k+1} \leftarrow Am$
end for
$As \leftarrow \bigcup_{i=1}^{n+1} As_i, Vs \leftarrow \bigcup_{i=1}^{n+1} Vs_i$
return As, Vs

Using different kinds of measures, we can obtain information from the graph structure and its content. A balanced mix of these measures will allow us to establish different criteria for the appropriate selection of categories, in order to include them in the final ontology.

There are several approaches which use Wikipedia to determine relatedness measures for terms. Let us briefly review the most representative.

WikiRelate! [Strube and Ponzetto, 2006] uses for the first time Wikipedia for relatedness measure computation. This proposal takes semantic relatedness measure traditionally used on WordNet and adapts them to Wikipedia. Given a pair of terms i and j, pages for the terms are recovered and a disambiguation process is conducted. Categories for the pages are searched in the category graph, and a relatedness measure is computed on them. The measures used in this approach include Rada, Wu-Palmer, Leacock-Chodorow, Resnik y Lesk.

In [Gabrilovich and Markovitch, 2007] *Explicit Semantic Analysis* (*ESA*) is presented. This approach represents text meaning in a high dimensionality space derived from Wikipedia. Using machine learning techniques, the meaning of a text is presented as a weighted vector of concepts extracted from Wikipedia. Similarity between texts is computed using cosine similarity on the concept vectors. This method presents a very good correlation with human judgements, but computation complexity is very high because each concept maps to a page in Wikipedia, so all pages must be processed and compared.

In [Milne and Witten, 2008] a novel technique called *Wikipedia Link Vector Model* (WLVM) was presented. This approach computes semantic relatedness between terms using Wikipedia links and pages titles, instead of exploiting text content.

Although very similar to these approaches our goal is a bit different. We want to select a sub-graph of Wikipedia category graph, relevant to the terms processed. In order to compare categories we must compute semantic relatedness as in the previous approaches. We want the process to be on line, which discards costly approaches such as ESA and WLVM. As in WikiRelate, we will experiment with well known similarity and relatedness measures, in order to perform category selection.

Category Selection using Measures

In order to generate the ontology using measures we perform the following steps:

- For each term in the AP-Structure verifying the syntactic rules presented in Section 3.2.4, get the category of the same name.

- If it is not possible to find a category for the term, search for a page containing the term in the title. For compound terms search for each of its components i.e. if no page is found for *Fuzzy Relational Database* search for *Fuzzy Database* and *Relational Database* (always conforming to the basic syntactic rules).

- Select the categories to which the pages are associated and use them as base categories or start categories. From these categories we start the exploration of the graph.

- While traversing the graph the categories in the path are compared. As we will see later, the comparison can be performed in different ways, but always return a value. The value must be greater than a threshold t previously set. Each evaluated category whose value it is greater than the threshold is included in the final ontology.
- The process is the same for each term. Once all terms are processed, all the subgraphs obtained are merged.

There are two important aspects that we have ignored until now, the way in which comparison is performed and the scope of the comparison. The particular way of comparing categories and the results yielded will depend on the measure employed, so we will deal with it when we present the different measures. Regarding the scope of the comparisons performed, that is, which categories to compare, we distinguish the three following scopes:

Comparison with the start category

This is the basic method, comparisons are performed with the start category while traversing the graph. The method used to explore the graph is the one described in Section 3.3.3. As we walk the graph towards the root each new category found is compared to the start category. If the value for the comparison is equal or greater than a previously set threshold t, the category is selected and added to the final result. If the comparison value is below the threshold the category not only is not added to the result, but the exploration of that branch stops.

The whole process of exploration and comparison is detailed in Algorithm 11. The function *category*() returns a comparable representation of the category associated to a category node. The representation will vary depending on the measure used. The function *compare*() returns the similarity between two category representations. The particular implementation of this function will depend on the measure. We will deal with category representation and comparison for each of the different measures in their own sections.

The main drawback of this comparison scope is the decreasing similarity of categories when the distance increases. As categories start becoming more general, they lose similarity with respect to the start category. However, it is possible that a related general category is connected to the start category through an intermediate category in a transitive way. Although this approach does not take this into account, the next one goes on that direction.

Comparison with the current category

This new method proposes to change the scope of comparisons in order to compare the new accepted categories to the ones close to them. If a new category is selected, that means it is relevant, and thus it acts as a new start category. Using this comparison scope, categories are compared only to their parent categories. This allows to update the context in each new step.

The whole process is described in Algorithm 12.

The drawback of this approach is that the context update can produce the selection of a general category. This general category is compared to other general categories, and so on;

Algorithm 11 Algorithm for Basic Exploration comparing with the start category

Require: The graph $G_W = (V, A, r)$, a set of start nodes $Vc = \{v_1, \ldots, v_n\}$ and a threshold t

Ensure: The set of nodes Vs and the set of arcs As

for all $v_l \in Vc$ **do**
  $C_1 \leftarrow category(v_l)$
  $Vs_0 \leftarrow \emptyset, As_0 \leftarrow \emptyset$
  $Vs_1 \leftarrow v_l, As_1 \leftarrow \emptyset$
  $k \leftarrow 2$
  for all $\{v_i \in Vs_{k-1} \wedge v_i \notin \bigcup_{j=0}^{k-2} Vs_j\}$ **do**
    if $\{a \in A, v \in V : init(a) = v_i \wedge ter(a) = v\}$ **then**
      $C_2 \leftarrow category(v)$
      if $compare(C_1, C_2) >= t$ **then**
        $As_k \leftarrow a, Vs_k \leftarrow v$
      end if
    end if
    $Vs_k \leftarrow [Vs_k \backslash Vs_{k-1}]$
    $k++$
  end for
  $As \leftarrow \bigcup_{i=1}^{k-1} As_i, Vs \leftarrow \bigcup_{i=1}^{k-1} Vs_i$
end for
return As, Vs

Algorithm 12 Algorithm for Basic Exploration comparing with the current category

Require: The graph $G_W = (V, A, r)$, a set of start nodes $Vc = \{v_1, \dots v_n\}$ and a threshold t

Ensure: A set of nodes Vs and a set of arcs As

for all $v_l \in Vc$ **do**
  $Vs_0 \leftarrow \emptyset, As_0 \leftarrow \emptyset$
  $Vs_1 \leftarrow v_l, As_1 \leftarrow \emptyset$
  $k \leftarrow 2$
  for all $\{v_i \in Vs_{k-1} \wedge v_i \notin \bigcup_{j=0}^{k-2} Vs_j\}$ **do**
    if $\{a \in A, v \in V : init(a) = v_i \wedge ter(a) = v\}$ **then**
      $C_1 \leftarrow category(v_i)$
      $C_2 \leftarrow category(v)$
      if $compare(C_1, C_2) >= t$ **then**
        $As_k \leftarrow a, Vs_k \leftarrow v$
      end if
    end if
    $Vs_k \leftarrow [Vs_k \backslash Vs_{k-1}]$
    $k++$
  end for
  $As \leftarrow \bigcup_{i=1}^{k-1} As_i, Vs \leftarrow \bigcup_{i=1}^{k-1} Vs_i$
end for
return As, Vs

selecting categories that maybe are related to the current category but have no relation at all with the start category. This can be solved comparing with all the categories previously selected that act as a context.

Comparison with the Aggregated Category

If the previous method took decisions on a local level, this new method uses a context to make decisions on a global level. Each category accepted is added to the context. Each time a new category is going to be added, it is compared to the aggregated category acting as a context. This way, the decision of selecting a category depends on all the categories previously selected.

The process is formally described in Algorithm 13.

Algorithm 13 Algorithm for Basic Exploration comparing with the aggregated category

Require: The graph $G_W = (V, A, r)$, a set of start nodes $Vc = \{v_1, \dots v_n\}$ and a threshold t

Ensure: A set of nodes Vs and a set of arcs As

for all $v_l \in Vc$ **do**
  $C_1 \leftarrow category(v_l)$
  $Vs_0 \leftarrow \emptyset, As_0 \leftarrow \emptyset$
  $Vs_1 \leftarrow v_l, As_1 \leftarrow \emptyset$
  $k \leftarrow 2$
  for all $\{v_i \in Vs_{k-1} \wedge v_i \notin \bigcup_{j=0}^{k-2} Vs_j\}$ **do**
    if $\{a \in A, v \in V : init(a) = v_i \wedge ter(a) = v\}$ **then**
      $C_2 \leftarrow category(v)$
      if $compare(C_1, C_2) >= t$ **then**
        $As_k \leftarrow a, Vs_k \leftarrow v$
        $C_1 \leftarrow C_1 \cup C_2$
      end if
    end if
    $Vs_k \leftarrow [Vs_k \backslash Vs_{k-1}]$
    $k++$
  end for
  $As \leftarrow \bigcup_{i=1}^{k-1} As_i, Vs \leftarrow \bigcup_{i=1}^{k-1} Vs_i$
end for
return As, Vs

The main problem with this approach is that each time a new category is added to the aggregated category, it becomes more general. Due to this increasing generality each time, it is easier to select a new category.

In general, the process to generate the ontology using measure based methods is the

following:

- Select the type of measure to use.
- Select a comparison scope.
- Select a comparison threshold value.
- The measure selected and the comparison scope do not change during the generation process.
- A start category for each term mus be determined.
- A path to the root is traversed from the start category.
- For each category found in the path:
  - Extract relevant information from the category found. This information depends on the measure used.
  - Extract relevant information from the category to compare. It changes depending on the comparison scope used.
  - Compare both categories using the measure.
  - If the value obtained is above a predefined threshold, the category found in the path is added to the solution.
  - If the category is not selected, exploration ends for that path. Other open paths for the same category can continue the exploration while the values obtained are above the threshold.

Next sections present well known similarity/relatedness measures in literature. We will study the effect of using them to generate ontologies from Wikipedia.

3.3.5 Statistical Indexes

Given that each category contains a set of pages, and each page is associated to a set of categories, we can consider each category a set of pages. Hence, pages are samples of the set defined by a category. We can apply statistical measures to two categories in order to obtain the similarity between them. This similarity is computed based on the pages contained in the category.

Let us review some of the most known statistical measures. We will define A and B as sets of Wikipedia pages associated to its corresponding categories a and b.

Jaccard Similarity Coefficient

Jaccard Similarity Coefficient was defined to study diversity in sample sets [Jaccard, 1901]. Jaccard index is defined as the size of the intersection set of the samples divided by the size of the union set of the samples:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}. \tag{3.2}$$

Similarly, the Jaccard distance is defined as the complement of the similarity index, and measures the diversity degree of sample sets.

$$J_\delta(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}. \tag{3.3}$$

Sorensen-Dice Similarity Coefficient

Sorensen Similarity Coefficient it is used to determine the similarity between two samples [Sorensen, 1948].

$$QS = \frac{2|A \cap B|}{|A| + |B|}. \tag{3.4}$$

This index is equivalent to the Dice index [Dice, 1945], thus we will refer to it as Sorensen-Dice index.

Mountford Index of Similarity

Mountford Index of Similarity [Mountford, 1962], is an index dependent of the sample set size.

$$M(A,B) = \frac{2|A \cap B|}{2|A||B| - (|A| + |B|)(|A \cap B|)}. \tag{3.5}$$

Overlap Index

The overlap index is a similarity measure related to Jaccard Similarity Coefficient which computes the overlap between two sample sets.

$$overlap(A,B) = \frac{|A \cap B|}{min(|A|,|B|)}. \tag{3.6}$$

If A is a subset of B or the other way around the index yields a value of 1.

Category selection using Statistical Indexes

In order to compare categories we characterize each category as a set of pages. Each page is represented by its title. The title is processed as a string identifier and hence it is not split into words. Figure 3.28 shows a comparison example between two categories C_1 and C_2, where each page has a set of pages $C_1 = \{p_a, p_b, p_c\}$ and $C_2 = \{p_b, p_d\}$. Each page has a title, the set of titles obtained from the pages in C_1 is labeled as A and the set of titles obtained from pages in C_2 is labeled B. When a statistical index is applied over the two sets A and B representing categories to compare, it returns a similarity value between both sets and hence between the two categories.

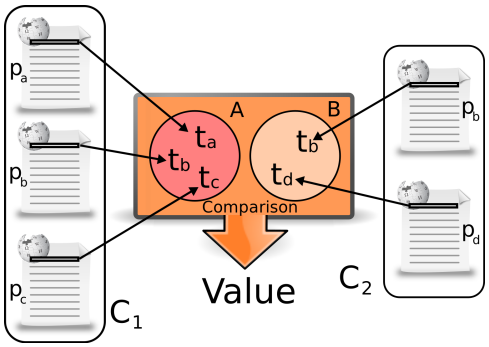


Figure 3.28: Category comparison using Statistical Indexes

Figure 3.29 shows the sub-tree of the category graph including all paths form the term *SQL*, comparing with the start category as depicted in Algorithm 11.

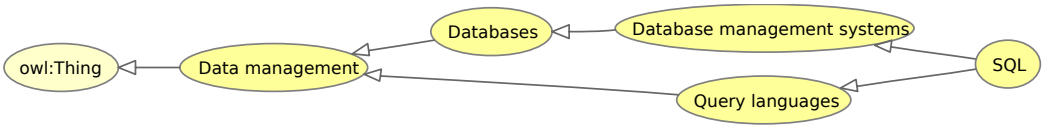


Figure 3.29: Result for the Similarity Measures

Table 3.4 contains results of similarity measures for a pair of categories C_1, C_2.

Table 3.4: Similarity values comparing with the start category

C_1	C_2	Jaccard	Sorensen	Mountford	Solapamiento
SQL	Database management systems	0,077777	0.144329	0.001898	0.197183
SQL	Query languages	0,061403	0.115702	0.002239	0.14
SQL	Databases	0,015384	0.030303	0.000304	0.056338
SQL	Data management	0,015384	0.030303	0.000285	0.070423

As results show, the set of categories selected is the same for all measures. We can see how as the distance to the start category increases the similarity decreases. It is important to stress that these measures are based only in the categories content, but we can see how

this behaviour is coherent with the topological measures previously reviewed. These results reinforce the idea that the farthest two categories are the less similar they are.

The results for comparison with the current category as depicted in Algorithm 12 are presented in Figure 3.30. In this case we can see that changing the scope of comparison we can move forward in the graph. However some leaps from a category to its parent cannot be achieved due to generality of the upper concept. Although the relation exists, as it is reflected in the structure, the leap from a particular concept to other more general some times cannot be done.

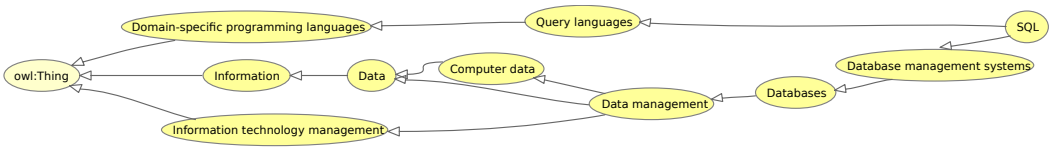


Figure 3.30: Results for similarity measures comparing with the current category

Table 3.5 shows values obtained for different indexes when using as comparison scope the current category.

Table 3.5: Similarity values comparing with the current category

C_1	C_2	Jaccard	Sorensen	Mountford	Overlap
SQL	SQL	1	1	1/0	1
SQL	Database management systems	0,077778	0,14433	0,001898	0,197183
SQL	Query languages	0,061404	0,115702	0,002239	0,14
Database management systems	Databases	0,071186	0,132911	0,001028	0,170732
Query languages	Data management	0,003247	0,006472	0,000078	0,02
Query languages	Domain-specific progr. lang.	0,016393	0,032258	0,000559	0,04
Databases	Data management	0,046296	0,088496	0,00044	0,103627
Databases	Computer data	0,004065	0,008097	0,000097	0,018519
Databases	Computer science	0	0	0	0
Domain-specific progr. lang.	Programming languages	0	0	0	0
Data management	Project management	0	0	0	0
Data management	Information technology management	0,016162	0,031809	0,000131	0,032787
Data management	Information retrieval	0	0	0	0
Data management	Computer data	0,029605	0,057508	0,000716	0,166667
Data management	Data	0,007435	0,01476	0,000705	0,166667
Information technology management	Information technology	0	0	0	0
Information technology management	Project management	0,006479	0,012876	0,000056	0,013514
Computer data	Data	0,03125	0,060606	0,003436	0,166667
Computer data	Computing	0	0	0	0
Data	Information	0,02439	0,047619	0,00295	0,083333
Information	Concepts	0	0	0	0

This results bring an interesting consideration, although the comparison is performed between parent-child categories the values obtained tend to decrease as we get closer to the root. This phenomenon is due to the increasing generality of categories as we approach the root node. This means that leaf nodes have more meaning than nodes closer to the root. Even similarity is higher for neighbour nodes far from the root regardless of having the same distance between them than other near to the root.

The problem now is that when we reach a general node it is very difficult to select it. The selection of nodes in lower parts of the graph is easier than in higher parts. Besides, we conduct the comparison based on the last category visited but we are ignoring all the history of previously selected categories.

In order to avoid the aforementioned problems, in Algorithm 13 we presented another scope of comparison based on the aggregation of the already accepted categories.

Table 3.6 shows the values obtained for the term *SQL* when using the aggregated category. We can see in the data that the values decrease as in previous methods, but now there are some categories which break this tendency.

Table 3.6: Similarity values comparing with the aggregated category

C_1	Jaccard	Sorensen	Mountford	Solapamiento
Query languages	0,061404	0,115702	0,002239	0,14
Database management systems	0,077273	0,14346	0,001416	0,149123
Data management	0,027897	0,05428	0,000241	0,059091
Domain-specific progr. lang.	0,003717	0,007407	0,000059	0,027027
Databases	0,065598	0,123119	0,000515	0,233161
Information technology management	0,01087	0,021505	0,000061	0,040984
Project management	0,003515	0,007005	0,00002	0,018018
Computer data	0,010373	0,020534	0,000223	0,185185
Information retrieval	0	0	0	0
Data	0,002053	0,004098	0,000189	0,166667
Programming languages	0	0	0	0
Computer science	0	0	0	0
Project management	0,003356	0,006689	0,000019	0,018018
Information technology	0,000995	0,001988	0,000033	0,03125
Computing	0	0	0	0
Information	0,00388	0,007729	0,000142	0,133333
Concepts	0	0	0	0

This approach presents a good behaviour at first as allows to explore the immediate vicinity and jump to other general categories. However this poses an inconvenient as the upper parts of the graph are highly connected. In this zone each new category added makes easier the acceptance of new categories, therefore it is possible to end up with a very big graph. In order to avoid this situation we must carefully select the threshold value.

The threshold value allows us to control the size of the final ontology. But we must be very careful as a low threshold value can induce a massive selection of categories located close to the root. This results in bigger ontologies with a lot of general non relevant classes.

In general we can say that statistical indexes provide low similarity values that are not easily interpretable by a human.

Taking into account all the data gathered we can see that statistical indexes do not provide enough discriminant power. Because of this we will analyze other measures that compute similarity taking into account semantic information encoded in the topological structure and category content characteristics.

3.3.6 Semantic Similarity Measures on Ontologies

Semantic similarity measures compute nearness or resemblance between two concepts in an ontology. The distance between two objects in an ontology is represented as a numerical value pointing out how far are two concepts in a given geometric space. Closer or similar concepts yield low distance values. If the relation between distance and similarity is sustained, it is possible to search for related elements that are not easily recognizable.

It is important to distinguish between semantic similarity and semantic relatedness. Semantic similarity represent certain degree of equality and is given by synonymy and hypernymy relations. The idea of semantic relatedness is more general and includes any

kind of lexical or functional association (including similarity) between a pair of words. In most of applications is enough to compute semantic relatedness, but it is important not to mix up both concepts.

Semantic similarity measures take as input two concepts and return a numeric quantification of the resemblance of those concepts. These measures are based usually on hierarchical relations defined in an ontology or taxonomy to which the concepts belong.

Ontology based measures tend to exploit topological information in terms of path lengths.

We review some of the most known similarity measures int literature.

Rada et al. Path-length Measure

Path-length measure [Rada et al., 1989] computes similarity between two concepts in an ontology as the number of nodes in the shortest path among them in a IS-A hierarchy. The formal definition of similarity between two concepts c_1 y c_2 is defined as:

$$sim_{path}(c_1, c_2) = 1/p, \tag{3.7}$$

where p is the number of nodes in the shortest path between the two hierarchy concepts. A disadvantage of this measure is that it ignores the global size of the hierarchy. Distance and hence similarity should be evaluated in the global context of the hierarchy. The next measure takes that into account.

Leacock-Chodorow Measure

Leacock-Chodorow measure [Leacock and Chodorow, 1998] defines similarity between two hierarchy concepts c_1 and c_2 as

$$sim_{lch}(c_1, c_2) = -log\left(\frac{p}{2 \cdot depth}\right), \tag{3.8}$$

where p is the number of nodes in the shortest path between the two hierarchy concepts and *depth* is the maximum depth of the hierarchy.

Wu-Palmer Measure

Wu-Palmer measure [Wu and Palmer, 1994] is a escalated measure that takes into account the depth of nodes and the depth of their *least common subsumer (LCS)*. This measure was reformulated in [Resnik, 1999] as follows:

$$sim_{wp}(c_1, c_2) = \frac{2 \cdot depth(lcs_{c_1,c_2})}{depth(c_1) + depth(c_2)}. \tag{3.9}$$

where *depth()* is a function which returns the depth of the node in the hierarchy.

Category Selection using Semantic Similarity Measures

The measures previously presented are semantic similarity measures and rely on the hypernym/hyponym structure of the graph. These measures do not use additional information from the categories, they only rely on information encoded in the category graph structure.

These measures are created to determine the similarity between any two nodes in an ontology. In our case these measures can be superseded by the exploratory method previously presented. Similarity measures do not provide meaningful information because while traversing the graph we ensure the topological proximity of concepts. Semantic similarity is ensured in the way we traverse the graph.

When using Rada path-length similarity between a node and its parents is 0.5. If we use Rada measure with a threshold of 0.5 and compare with the start node, it is equivalent to use the path length exploratory method of Section 3.3.3 with $n = 2$. In general Rada measure is equivalent to the method when t is selected as $t = sim_{path}(c_1, c_2) = 1/n$ where n is the parameter of the path length basic method. If the comparison is performed using the current category/node the value for Rada is always maximum (with a value of 0.5) and thus the method is equivalent to the basic method presented in Section 3.3.3. It is not possible to use the comparison to the aggregate category, as it is not possible to represented a topological aggregation of nodes. It is possible though to chose a representative or centroid of the set of selected nodes, but we discard this possibility as the use of topological information alone it is not enough for our purposes.

As the depth parameter is fixed, we find an equivalent situation in Leacock-Chodorow. Similarity is very high for adjacent nodes and the situation is equivalent to the Rada case, but the values of similarity provided are different.

The Wu-Palmer similarity measure can be adapted to our way of traversing the graph. As we compare a node with its parent, the LCS of both is the parent node. So it is possible to rewrite the measure as follows:

$$sim_{wp}(c_1, c_2) = \frac{2 \cdot depth(c_2)}{depth(c_1) + depth(c_2)}. \tag{3.10}$$

Table 3.7 shows the similarity values obtained from the different measures when comparing to the start category.

The first thing to capture our attention are the non normalized values obtained for the Leacock-Chodorow measure. This situation is produced by the logarithmic scaling used in the measure. The Rada measure ranges between 1 (bot categories are the same) and a lower bound of 0. The measure tends to 0 as the distance between nodes increases. Regarding Wu-Palmer measure, it reaches a value of 1 when both nodes are the same, as with Rada. A value of 0 is obtained when both nodes have no common ancestor (their LCS is the root).

The data obtained confirm that, although the topological characterization of the nodes is equivalent, their semantics are not. These measures cannot express more semantics than the one implicit in the category graph structure. A clear example can be seen with terms *Databases* and *Data Management*; both are at the same distance from *SQL*, and yield identical similarity values. But we can see that *Databases* is clearly semantically closer to

Table 3.7: Similarity values obtained comparing with the start category

C_1	C_2	Leacock-Chodorow	Wu-Palmer	Path Length
SQL	SQL	2,995732	1	1
SQL	Query languages	2,302585	0,6	0,5
SQL	Database management systems	2,302585	0,888889	0,5
SQL	Data management	1,89712	0,666667	0,333333
SQL	Domain-specific progr. lang.	1,89712	0,6	0,333333
SQL	Databases	1,89712	0,666667	0,333333
SQL	Data	1,386294	0,6	0,2
SQL	Project management	1,609438	0,6	0,25
SQL	Computer data	1,609438	0,6	0,25
SQL	Information technology management	1,609438	0,666667	0,25
SQL	Information retrieval	1,386294	0	0,2
SQL	Programming languages	1,609438	0,666667	0,25
SQL	Computer science	1,609438	0,666667	0,25
SQL	Information	1,386294	0,6	0,2
SQL	Management	1,609438	0,666667	0,25
SQL	Product development	1,386294	0,545455	0,2
SQL	Production and manufacturing	1,203973	0	0,166667
SQL	Computing	1,609438	0,666667	0,25
SQL	Information technology	1,609438	0,666667	0,25
SQL	Formal sciences	1,386294	0,6	0,2
SQL	Applied sciences	1,386294	0	0,2
SQL	Information	1,386294	0,6	0,2
SQL	Information science	1,203973	0	0,166667
SQL	Computer languages	1,609438	0,6	0,25
SQL	Programming language topics	1,386294	0,6	0,2
SQL	Computer programming	1,386294	0,6	0,2
SQL	Digital technology	1,386294	0,6	0,2
SQL	Interdisciplinary fields	1,386294	0	0,2
SQL	Abstraction	1,203973	0,545455	0,166667
SQL	Scientific disciplines	1,203973	0	0,166667
SQL	Formalism (deductive)	1,203973	0,545455	0,166667
SQL	Academic disciplines	1,203973	0,545455	0,166667
SQL	Applied disciplines	1,203973	0,545455	0,166667
SQL	Concepts	1,609438	0,666667	0,25
SQL	Business	1,609438	0,666667	0,25
SQL	Business economics	1,386294	0,444444	0,2
SQL	Accountability	1,386294	0,4	0,2
SQL	Product management	1,203973	0,545455	0,166667
SQL	Industry	1,386294	0	0,2
SQL	Science studies	1,203973	0	0,166667
SQL	Library and information science	1,203973	0,545455	0,166667
SQL	Notation	1,386294	0,545455	0,2
SQL	Constructed languages	1,386294	0,6	0,2
SQL	Software engineering	1,609438	0,666667	0,25

SQL than *Data Management*.

Let us change the comparison scope and compare with the current category. We can see the results in Table 3.8.

This new set of data results highlights all the drawbacks for these methods. The lack of semantic discrimination is clearly visible in the measures of Leacock-Chodorow and Rada.

Rada measure always yield a value of 0.5 because that is the value when comparing a node with its parent. The same occurs with the Leacock-Chodorow measure although the value in this case is 2,302585 because as we stated earlier it is not normalized. The variance in Wu-Palmer method is higher due to the LCS calculation. In order to speed-up processing the graph is converted to a tree and the paths to the root are pre-computed and stored in a file. This file is used to calculate LCSs. This can cause that a pair of nodes with a direct connection are not connected in the pre-processed file (they are not part of the same shortest path), causing some inconsistencies in the calculation of the LCS. Unfortunately this file it is necessary in order to produce results in a short timespan, so we have to accept this minor glitches.

It is clear that changing the comparison scope to the current category for semantic similarity measures it is not useful. However it is been a good way to empirically prove the small discriminant power of these measures. At this point, it is clear that, in order to compare effectively two categories, we must take into account their content.

3.3.7 Information Content Measures

Measures based on the distance between concepts in an ontology, do not take into account that the semantics of a link between concepts can change depending on where the concepts are. A link between two very specific concepts should not be interpreted in the same way as a link between two general concepts. In the latter case, probably the similarity is less.

In order to avoid this drawback, there are methods that use topological information in addition to the information content inherent in a concept. Information content based measures take into account the geometric structure of the hierarchy and try to determine the contribution of each concept to the ontology. General concepts closer to the root are less informative than leaf concepts whose semantics are defined precisely. In order to determine this contribution external sources are used. Let us briefly review the most known information content measures in literature.

Resnik Information Content

In [Resnik, 1995] a new similarity measure over *IS-A* taxonomies is presented. It introduces the idea of information content. Resnik information content measure combines information obtained from the taxonomic structure with probabilistic estimations for concepts obtained empirically from a huge text corpus. Similarity is modeled as the degree in which concepts share information and it is computed as the information content of its lowest common ancestor (LCS).

$$sim_{res}(c_1, c_2) = IC(lcs_{c_1,c_2}). \tag{3.11}$$

Table 3.8: Similarity values obtained comparing with the current category

C_1	C_2	Leacock-Chodorow	Wu-Palmer	Path Length
SQL	SQL	2,995732	1	1
SQL	Query languages	2,302585	0,6	0,5
SQL	Database management systems	2,302585	0,888889	0,5
Database management systems	Databases	2,302585	0,75	0,5
Query languages	Data management	2,302585	0,888889	0,5
Query languages	Domain-specific progr. lang.	2,302585	0,6	0,5
Databases	Computer data	2,302585	0,666667	0,5
Databases	Computer science	2,302585	0,75	0,5
Databases	Data management	2,302585	0,75	0,5
Data management	Information retrieval	2,302585	0	0,5
Data management	Data	2,302585	0,666667	0,5
Data management	Project management	2,302585	0,666667	0,5
Data management	Computer data	2,302585	0,666667	0,5
Data management	Information technology management	2,302585	0,75	0,5
Domain-specific progr. lang.	Programming languages	2,302585	0,888889	0,5
Computer data	Computing	2,302585	0,888889	0,5
Computer data	Data	2,302585	0,6	0,5
Computer science	Formal sciences	2,302585	0,666667	0,5
Computer science	Applied sciences	2,302585	0	0,5
Computer science	Computing	2,302585	0,75	0,5
Information retrieval	Information science	2,302585	0,888889	0,5
Data	Information	2,302585	0,888889	0,5
Project management	Production and manufacturing	2,302585	0	0,5
Project management	Management	2,302585	0,666667	0,5
Project management	Product development	2,302585	0,909091	0,5
Information technology management	Project management	2,302585	0,666667	0,5
Information technology management	Information technology	2,302585	0,75	0,5
Programming languages	Programming language topics	2,302585	0,666667	0,5
Programming languages	Computer programming	2,302585	0,666667	0,5
Programming languages	Computer science	2,302585	0,75	0,5
Programming languages	Computer languages	2,302585	0,666667	0,5
Computing	Digital technology	2,302585	0,666667	0,5
Computing	Information technology	2,302585	0,75	0,5
Formal sciences	Formalism (deductive)	2,302585	0,909091	0,5
Formal sciences	Interdisciplinary fields	2,302585	0	0,5
Formal sciences	Academic disciplines	2,302585	0,545455	0,5
Formal sciences	Abstraction	2,302585	0,545455	0,5
Formal sciences	Scientific disciplines	2,302585	0	0,5
Applied sciences	Applied disciplines	2,302585	0	0,5
Applied sciences	Scientific disciplines	2,302585	0,857143	0,5
Information science	Information	2,302585	0	0,5
Information science	Science studies	2,302585	0,857143	0,5
Information science	Library and information science	2,302585	0	0,5
Information science	Interdisciplinary fields	2,302585	0,571429	0,5
Information	Concepts	2,302585	0,857143	0,5
Production and manufacturing	Industry	2,302585	0,888889	0,5
Management	Business	2,302585	0,75	0,5
Management	Accountability	2,302585	0,444444	0,5
Management	Business economics	2,302585	0,5	0,5
Management	Applied sciences	2,302585	0	0,5
Product development	Production and manufacturing	2,302585	0	0,5
Product development	Product management	2,302585	0,909091	0,5
Programming language topics	Software engineering	2,302585	0,888889	0,5
Programming language topics	Computer languages	2,302585	0,6	0,5
Computer programming	Software engineering	2,302585	0,666667	0,5
Computer programming	Computing	2,302585	0,888889	0,5
Computer languages	Constructed languages	2,302585	0,6	0,5
Computer languages	Notation	2,302585	0,545455	0,5
Computer languages	Computing	2,302585	0,888889	0,5

In the original article the information content (IC) is computed using word frequencies from a text corpus.

$$IC(c) = -log\left(\frac{freq(c)}{freq(root)}\right), \tag{3.12}$$

where $freq(c)$ is the concept frequency c, and $freq(root)$ is the frequency for the root concept. The concept frequency is obtained in the taxonomy from the frequencies computed on the *Brown Corpus of American English*. This corpus contains more than a million words, having fragments of news and fiction novels. It is important to point out that, when the frequency count is performed, each term counts on its concept and in all concepts which subsume it.

In [Seco et al., 2004] a variant to Resnik information content is presented. This variant relies only on WordNet for information content calculation without using any external corpus.

$$IC_{wn}(c) = 1 - \frac{log(hypo(c) + 1)}{log(max_{wn})}, \tag{3.13}$$

where $hypo(c)$ returns the number of hyponyms (concepts subsumed) for the concept c and max_{wn} is a constant with the total number of concepts in the taxonomy. This reformulation of the information content is equivalent to the Resnik formulation if the frequency of each term is 1. Both definitions yield similar values, which means that information about frequency in a external corpus it is not as important as topological information, particularly information concerning the depth of the concept in the taxonomy.

Lin Similarity Measure

Resnik information content is computed on the LCS of two concepts. The problem is that many concepts can share the same LCS and that does not mean that they are similar in the same degree. To solve this problem [Lin, 1998] proposes to scale information content of the LCS by the information content of the subsumed concepts. Lin uses a ratio, computing similarity between two concepts as:

$$related_{lin}(c_1, c_2) = \frac{2 \times IC(lcs(c_1, c_2))}{IC(c_1) + IC(c_2)}, \tag{3.14}$$

where $lcs(c_1, c_2)$ is the least common subsumer and IC returns the information content for a concept.

In [Patwardhan et al., 2003] this measure is interpreted as 2 times the information content of the intersection of two concepts, divide by the sum of the information content of the concepts. This is analogous to the Sorensen-Dice coefficient seen in Section 3.3.5. Moreover, we postulate that it is an adaptation of the Wu-Palmer measure defined by Resnik previously shown in Section 3.3.6, only that it changes the use of topological measures for the use of information content.

Jiang-Conrath Distance

Another possible alternative to the Resnik measure was proposed in [Jiang and Conrath, 1997]. In this case Jiang and Conrath scale the information content of the LCS with respect to their subsumed concepts using subtraction. Semantic distance for the concepts c_1 and c_2 is computed as follows:

$$dist_{jcn}(c_1, c_2) = IC(c_1) + IC(c_2) - 2 \times IC(lcs(c_1, c_2)). \tag{3.15}$$

After conversion to a relatedness measure and normalization the Jiang-Conrath measure is reformulated as follows:

$$related_{jcn}(c_1, c_2) = \frac{2 - (IC(c_1) + IC(c_2) - 2 \times IC(lcs(c_1, c_2)))}{2}. \tag{3.16}$$

Category Selection using Information Content Measures

Information content based measures take into account the topological structure of the hierarchy and the information about term frequencies extracted from external sources. This use of frequencies establishes the contribution of each node to the more global concept, that is the hierarchy root. As the Seco measure highlights, the importance of term frequencies is not as determinant as the position of the nodes in the hierarchy.

As in all measure based methods, we will determine the start categories for each term, and then traverse a path to the root comparing with the categories found using the scopes presented earlier. In this case it is not possible to use the comparison with the aggregated category, because we cannot represent a topological aggregation of categories. However, it would be possible to chose a representative or centroid, but for now we will adhere to the process conducted until now.

In order to compute values for these measures we use the JWPL (Java Wikipedia Library) API [Zesch and Gurevych, 2010]. JWPL is an open source application programming interface that enables Wikipedia information access. This API can build a hierarchy from the Wikipedia category graph in order to compute the values of function $hypo(c)$ of Seco Information Content. The main drawback for these measures it is the need to adapt the graph to a hierarchical structure removing cycles and redundant paths. The way to compute the hierarchy is computing the shortest paths from the leaf nodes to the root, then from the remainder nodes to the root. To improve efficiency it is assumed that all nodes in a shortest path have no other shortest path apart from the one to which they belong. The heuristic starts with leaf nodes because that provides the shortest paths with a higher number of nodes. We must remember that the number of leaf nodes in the graph we are working with is about 381000. The use of the heuristic for the hierarchy construction has an unwanted side effect, in some cases nodes having a LCS in the graph, can be in different non-overlapping paths in the hierarchy, thus losing that connection. This poses a great drawback, as some measures rely on the computation of a LCS, but other solutions prove to be time consuming and ineffective. The creation of the hierarchy is mandatory as these measures compute subsumption hierarchies that cannot contain cycles or multiple paths to a single node.

With our way of traversing the graph, from leaf nodes towards the root, every node visited will be the LCS of itself and the start node. This avoids the calculation of the LCS using the hierarchy, which is incomplete, and saves some time. This allows to reformulate Resnik $sim_{res'}(c_1, c_2) = IC(c_2)$ as the way of traversing the graph guarantees $lcs(c_1, c_2) = c_2$.

Table 3.9 shows the values obtained by the IC measures using the basic exploration and comparison with the start category. Evaluations have been performed using Seco definition of information content, to avoid the use of an external corpus. The table shows the values for the IC of the categories and their LCS, a value of 0 means that no LCS is found for the two categories. This case is very frequent and it is a direct consequence of the use of the computed hierarchy as we explained before. We provide results for two versions of the measure, the original using the computed hierarchy, and our version using $lcs(c_1, c_2) = c_2$.

The first thing to capture our attention is that the heuristic used to generate the paths to the root destroys most of the relations in the graph, this is reflected in the cases where the *lcs* cannot be computed. We observe that applying the restriction $lcs(c_1, c_2) = c_2$ it is possible to obtain more values, but the difference it is not significant.

It is interesting how Resnik measure gives contra-intuitive results as the comparison of category *SQL* with itself yields a similarity value of 0,8655, while if it is compared with *Formalism* the value obtained is 0.976, even when the latter is more general (and thus its IC should be less). Moreover the IC value of the comparison of a category with itself should be 1.

In the case of values obtained for Lin, *Formalism* yields a value of 1,0292, this only can mean that when breaking cycles the category ended up being closer to a leaf node than *SQL*. The same goes for the Jiang-Conrath measure where values greater than 1 are obtained.

The initial results do not seem intuitive, it would be necessary further study to determine if this kind of measures can be useful when applied to Wikipedia.

We change the scope to test if the results obtained improve with respect to the previous approach. Table 3.10 shows results for information content measures changing the comparison scope to the current category.

The absence of IC values for the *LCS* is again a notorious problem. This measures depend heavily on the hierarchy topology. If we change the topology to ease processing, the new values are biased and lose relevance. Using the measures on the original structure is not a possibility either, as it is pervaded with cycles which makes difficult the IC computation.

In conclusion, these measures are not easily adaptable to Wikipedia category graph. The measure providing consistent results seems to be Jiang-Conrath, when using comparison with the start category. A change in the implementation of the hierarchy generation can solve these problems, but the computational cost and the time needed to generate the hierarchy are excessive.

3.3.8 Information Retrieval Based Methods

This section focuses on the comparison of categories based on their textual content using information retrieval techniques. The basic idea is to obtain the text associated to the

Table 3.9: Similarity values using IC measures and comparing to the start category

C_1	C_2	IC(C_1)	IC(C_2)	IC(lcs)	Resnik	Res. IC(C_2)	Lin	Lin IC(C_2)	J-C	J-C IC(C_2)
SQL	SQL	0,8655	0,8655	0,8655	0,8655	0,8655	1	1	1	1
SQL	Query languages	0,8655	0,8201	0	0	0,8201	0	0,973	0,1572	0,9773
SQL	Database management systems	0,8655	0,676	0,676	0,676	0,676	0,8771	0,8771	0,9052	0,9052
SQL	Data management	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Domain-specific progr. lang.	0,8655	0,6112	0	0	0,6112	0	0,8277	0,2616	0,8728
SQL	Databases	0,8655	0,4511	0	0	0,4511	0	0,6852	0,3417	0,7928
SQL	Information technology manag.	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Project management	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Information retrieval	0,8655	0,3496	0	0	0,3496	0	0,5754	0,3924	0,742
SQL	Computer data	0,8655	0,1097	0	0	0,1097	0	0,225	0,5124	0,6221
SQL	Data	0,8655	0,1097	0	0	0,1097	0	0,225	0,5124	0,6221
SQL	Programming languages	0,8655	0,4908	0	0	0,4908	0	0,7237	0,3218	0,8126
SQL	Computer science	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Information technology	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Management	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Production and manufacturing	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Product development	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Information science	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Computing	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Information	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Programming language topics	0,8655	0,451	0	0	0,451	0	0,6851	0,3417	0,7927
SQL	Computer languages	0,8655	0,4018	0	0	0,4018	0	0,6341	0,3663	0,7681
SQL	Computer programming	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Formal sciences	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Applied sciences	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Accountability	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Business	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Business economics	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Industry	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Product management	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Science studies	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Interdisciplinary fields	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Library and inf. science	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Digital technology	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Concepts	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Software engineering	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Notation	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Constructed languages	0,8655	0,3829	0	0	0,3829	0	0,6134	0,3758	0,7587
SQL	Formalism (deductive)	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Abstraction	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Scientific disciplines	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Academic disciplines	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Applied disciplines	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Social philosophy	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Corruption	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Core issues in ethics	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Society	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Economics	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Technology	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Branding	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Strategic management	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Science	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Libraries	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Technology by type	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Electronics	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Numbers	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Thought	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Systems engineering	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Conceptual systems	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Scientific modeling	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Communication	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Information systems	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Languages	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Theories of deduction	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Formalism	0,8655	0,9176	0	0	0,9176	0	1,0292	0,1084	1,026
SQL	Innovation	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Structure	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Creativity	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Problem solving	0,8655	0,7919	0	0	0,7919	0	0,9556	0,1713	0,9632
SQL	Academia	0,8655	0	0	0	0	0	0	0,5672	0,5672
SQL	Knowledge	0,8655	0	0	0	0	0	0	0,5672	0,5672

Table 3.10: Similarity values using IC measures and comparing to the current category

C_1	C_2	IC(C_1)	IC(C_2)	Resnik	R.IC(C_2)	Lin	L.IC(C_2)	J-C	JC.IC(C_2)
SQL	SQL	0,8655	0,8655	0,8655	0,8655	1	1	1	1
SQL	Database manag. syst.	0,8655	0,676	0,676	0,676	0,8771	0,8771	0,9052	0,9052
SQL	Query languages	0,8655	0,8201	0	0,8201	0	0,973	0	0,9773
Database manag. syst.	Databases	0,676	0,4511	0	0,4511	0	0,8005	0	0,8875
Query languages	Data management	0,8201	0	0	0	0	0	0,59	0,59
Query languages	Domain-specif. progr. lang.	0,8201	0,6112	0	0,6112	0	0,854	0	0,8955
Databases	Computer data	0,4511	0,1097	0	0,1097	0	0,3913	0	0,8293
Databases	Computer science	0,4511	0	0	0	0	0	0	0,7744
Databases	Data management	0,4511	0	0	0	0	0	0	0,7744
Data management	Information retrieval	0	0,3496	0	0,3496	0	0	0	1,1748
Data management	Data	0	0,1097	0	0,1097	0	0	0	1,0548
Data management	Project management	0	0	0	0	0	0	0	0
Data management	Computer data	0	0,1097	0	0,1097	0	0	0	1,0549
Data management	Information tech. manag.	0	0	0	0	0	0	0	0
Domain-specif. progr. lang.	Programming languages	0,6112	0,4908	0,4908	0,4908	0,8908	0,8908	0,9398	0,9398
Computer data	Computing	0,1097	0	0	0	0	0	0	0
Computer data	Data	0,1097	0,1097	0	0,1097	0	0,9999	0	1
Computer science	Formal sciences	0	0	0	0	0	0	0	0
Computer science	Applied sciences	0	0	0	0	0	0	0	0
Computer science	Computing	0	0	0	0	0	0	0	0
Information retrieval	Information science	0,3496	0	0	0	0	0	0	0
Data	Information	0,1097	0	0	0	0	0	0,9452	0,9452
Project management	Prod. and manufacturing	0	0	0	0	0	0	0	0
Project management	Management	0	0	0	0	0	0	0	0
Project management	Product development	0	0	0	0	0	0	0	0
Information tech. manag.	Project management	0	0	0	0	0	0	0	0
Information tech. manag.	Information technology	0	0	0	0	0	0	0	0
Programming languages	Programming lang. topics	0,4908	0,451	0	0,451	0	0,9577	0	0,9801
Programming languages	Computer programming	0,4908	0	0	0	0	0	0	0
Programming languages	Computer science	0,4908	0	0	0	0	0	0	0,7546
Programming languages	Computer languages	0,4908	0,4018	0	0,4018	0	0,9003	0	0,9555
Computing	Digital technology	0	0	0	0	0	0	0	0
Computing	Information technology	0	0	0	0	0	0	0	0
Formal sciences	Formalism (deductive)	0	0	0	0	0	0	0	0
Formal sciences	Interdisciplinary fields	0	0	0	0	0	0	0	0
Formal sciences	Academic disciplines	0	0	0	0	0	0	0	0
Formal sciences	Abstraction	0	0	0	0	0	0	0	0
Formal sciences	Scientific disciplines	0	0	0	0	0	0	0	0
Applied sciences	Applied disciplines	0	0	0	0	0	0	0	0
Applied sciences	Scientific disciplines	0	0	0	0	0	0	0	0
Information science	Information	0	0	0	0	0	0	0	0
Information science	Science studies	0	0	0	0	0	0	0	0
Information science	Library and inf. science	0	0	0	0	0	0	0	0
Information science	Interdisciplinary fields	0	0	0	0	0	0	0	0
Information	Concepts	0	0	0	0	0	0	0	0
Prod. and manufacturing	Industry	0	0	0	0	0	0	0	0
Management	Business	0	0	0	0	0	0	0	0
Management	Accountability	0	0	0	0	0	0	0	1
Management	Business economics	0	0	0	0	0	0	0	1
Management	Applied sciences	0	0	0	0	0	0	0	0
Product development	Prod. and manufacturing	0	0	0	0	0	0	0	0
Product development	Product management	0	0	0	0	0	0	0	0
Programming lang. topics	Software engineering	0,451	0	0	0	0	0	0	0
Programming lang. topics	Computer languages	0,451	0,4018	0	0,4018	0	0,9423	0	0,9754
Computer programming	Software engineering	0	0	0	0	0	0	0	0
Computer programming	Computing	0	0	0	0	0	0	0	0
Computer languages	Constructed languages	0,4018	0,3829	0	0,3829	0	0,9758	0	0,9905
Computer languages	Notation	0,4018	0	0	0	0	0	0	0,7991
Computer languages	Computing	0,4018	0	0	0	0	0	0	0
Digital technology	Technology by type	0	0	0	0	0	0	0	0
Digital technology	Electronics	0	0	0	0	0	0	0	0
Digital technology	Numbers	0	0	0	0	0	0	0	0
Formalism (deductive)	Theories of deduction	0	0	0	0	0	0	0	0
Formalism (deductive)	Formalism	0	0,9176	0	0,9176	0	0	0	0
Interdisciplinary fields	Science	0	0	0	0	0	0	1	1
Interdisciplinary fields	Academic disciplines	0	0	0	0	0	0	0	0
Academic disciplines	Academia	0	0	0	0	0	0	0	0
Abstraction	Structure	0	0	0	0	0	0	0	0
Abstraction	Problem solving	0	0,7919	0	0,7919	0	0	0	1,396
Abstraction	Innovation	0	0	0	0	0	0	0	1
Abstraction	Thought	0	0	0	0	0	0	0	1
Abstraction	Creativity	0	0	0	0	0	0	0	1
Scientific disciplines	Science	0	0	0	0	0	0	0	1
Applied disciplines	Knowledge	0	0	0	0	0	0	0	0
Applied disciplines	Academic disciplines	0	0	0	0	0	0	0	0
Science studies	Science	0	0	0	0	0	0	0	0
Science studies	Interdisciplinary fields	0	0	0	0	0	0	0	0
Library and inf. science	Libraries	0	0	0	0	0	0	0	0
Library and inf. science	Interdisciplinary fields	0	0	0	0	0	0	0	0
Concepts	Thought	0	0	0	0	0	0	0	0
Industry	Business	0	0	0	0	0	0	0	0
Industry	Technology	0	0	0	0	0	0	0	0
Business	Society	0	0	0	0	0	0	0	0

pages in the category, and compare two categories by their content.

There are two possible sources to extract textual data from a page $p = (t, c, L, LC)$, page title t or page content c. The page content includes the text of the page and the title.

Therefore we have two possibilities:

- Process text in page titles.
- Process text contained in the pages.

The comparison scopes are the three previously presented in Section 3.3.4:

- *Comparison with the start category.* Compares the terms extracted from the evaluated category with those extracted from the start category.
- *Comparison with the current category.* Compares the terms extracted from a category with those of its parent category.
- *Comparison with the aggregated category.* Compares the terms from the evaluated category with a aggregated category containing all the terms included in the categories selected to this point.

Categories whose relatedness is above a predefined threshold are selected. As comparison measure we use cosine similarity [Zobel and Moffat, 1998].

One of the most used representations for texts in information retrieval is the *vector space model*. This model encodes documents or texts as a vector of terms. To compare two documents we just need to apply cosine similarity on both vectors.

In this case we use the term vector to represent a category and the codification is performed as in [Manning et al., 2008].

The term vectors can be computed from the pages titles or from pages content, of those pages contained in a given category. Once the vectors are computed we apply cosine similarity following this formula:

$$sim_{cos}(c_1, c_2) = \frac{\vec{V}(c_1) \cdot \vec{V}(c_2)}{|\vec{V}(c_1)||\vec{V}(c_2)|} \tag{3.17}$$

where numerator represent dot product of vectors $\vec{V}(c_1)$ y $\vec{V}(c_2)$ and denominator is the product of their euclidean distances.
The dot product of two vectors $\vec{x} \cdot \vec{y}$ is defined as $\sum_{i=1}^{M} x_i y_i$.
For a text vector $\vec{V}(c)$ of a given category c, with M components $\vec{V}_1(c), \ldots, \vec{V}_M(c)$. the euclidean distance of c is defined as $\sqrt{\sum_{i=1}^{M} \vec{V}_i^2(c)}$.
The effect of the denominator in Equation 3.17 is the length normalization of the vectors $\vec{V}(c_1)$ y $\vec{V}(c_2)$ to unit vectors $\vec{v}(c_1) = \vec{V}(c_1)/|\vec{V}(c_1)|$ y $\vec{v}(c_2) = \vec{V}(c_2)/|\vec{V}(c_2)|$.

We must select an appropriate weighting schema for the vectors. The most used weighting schema in information retrieval is *tf-idf* (*term frequency-inverse term frequency*).

Tf-idf is defined as the product of a term frequency multiplied by the inverse frequency of the document. In our case instead of having documents we have categories.

$$\text{tf-idf}_{t,c} = \text{tf}_{t,c} \times \text{idf}_t. \tag{3.18}$$

Term frequency $\text{tf}_{t,c}$ is the number of times a certain term t appears in a category c. The frequency is normalized by the document length (in our case the total length of the contents in the pages of the category):

$$\text{tf}_{t,c} = \frac{n_{t,c}}{\sum_k n_{k,c}}, \tag{3.19}$$

where numerator represents the number of times that the term t appears in the text of category c. The denominator is the number of terms in the text of category c. This way term frequency is normalized by the text length.

It is possible to normalize frequency values to keep them in the interval $[0, 1]$. This is done changing the formulation of $\text{tf}_{t,c}$ as follows:

$$\text{wf}_{t,c} = \frac{n_{t,c}}{max_l\ n_{l,c}}, \tag{3.20}$$

where the numerator represents the number of time the t appears in the text for category c, and the denominator is the maximum frequency of the terms in the text.

With respect to idf_t, the inverse term frequency is defined as:

$$\text{idf}_t = \log \frac{N}{1 + \text{df}_t}. \tag{3.21}$$

where N is the total number of categories for which we are computing vector weights, and df_t is the frequency of term t in the categories, it represents the number of texts on which the term appear in a particular category. We add 1 to df_t, to avoid division by zero in those cases when the term does not appear in texts. The base for the logarithm is 10, but that do not changes results.

The idf_t value represents the importance of a term in the text collection, in our case, the relevance of a term to the set of categories used to compute the weights.

The *tf-idf* weighting schema is particularly suitable for document retrieval using textual search. However in this case the application is more oriented towards document comparison, as we do not want to retrieve categories using text search, as this would imply the processing and computation of *tf-idf* weights for all categories in the graph which implies a high computational cost.

We just want to compare categories in the same way text documents are compared. To do this it is not necessary to process all textual content available in Wikipedia. We use normalized term frequency $\text{wf}_{t,c}$ to compute weight vectors.

Category Selection based on Information Retrieval

For category selection using cosine similarity, we start using basic exploration and as comparison scope we use comparison with the start category. We include the results for the comparison using the title, the content, and their stemmed versions.

Using this configuration we get the results shown in Table 3.11.

Table 3.11: Similarity values using cosine similarity and comparing with the start category

C_1	C_2	Title	Title Stem	Content	Content Stem
SQL	SQL	1	1	1	1
SQL	Query languages	0,371164	0,367437	0,827852	0,827352
SQL	Database management systems	0,304376	0,328664	0,677177	0,707665
SQL	Data management	0,252432	0,261008	0,48218	0,524522
SQL	Domain-specific progr. lang.	0,143595	0,143456	0,391762	0,438828
SQL	Databases	0,259525	0,285854	0,607517	0,637537
SQL	Information technology management	0,080488	0,087887	0,331229	0,375797
SQL	Project management	0,045397	0,054935	0,220149	0,267638
SQL	Information retrieval	0,026042	0,034863	0,28368	0,330046
SQL	Computer data	0,15534	0,146782	0,37283	0,407454
SQL	Data	0,183383	0,18583	0,319919	0,331589
SQL	Programming languages	0,069589	0,069135	0,230309	0,282883
SQL	Computer science	0,015965	0,016131	0,235931	0,259773
SQL	Information technology	0	0	0,174878	0,21601
SQL	Management	0,059468	0,064298	0,262228	0,307915
SQL	Production and manufacturing	0,03604	0,048943	0,302411	0,338459
SQL	Product development	0,007698	0,012747	0,245017	0,289058
SQL	Information science	0,028382	0,032299	0,284907	0,337122
SQL	Computing	0,024714	0,021124	0,346479	0,357275
SQL	Information	0,008995	0,01619	0,192546	0,239355
SQL	Programming language topics	0,112747	0,134682	0,355021	0,401475
SQL	Computer languages	0,143208	0,150062	0,444754	0,488037
SQL	Computer programming	0,053736	0,056985	0,418974	0,444142
SQL	Formal sciences	0,051333	0,050225	0,312836	0,343224
SQL	Applied sciences	0,015462	0,019024	0,266593	0,294317
SQL	Accountability	0	0	0,127105	0,149626
SQL	Business	0,045087	0,046779	0,248313	0,285035
SQL	Business economics	0,012329	0,017822	0,242244	0,268359
SQL	Industry	0,008219	0,008447	0,25605	0,271739
SQL	Product management	0,034662	0,03948	0,22693	0,258233
SQL	Science studies	0	0,004945	0,180037	0,21213
SQL	Interdisciplinary fields	0,003255	0,00851	0,213505	0,241849
SQL	Library and information science	0,042771	0,052655	0,24931	0,283882
SQL	Digital technology	0,016294	0,018211	0,241216	0,285712
SQL	Concepts	0,010849	0,014472	0,204411	0,223563
SQL	Software engineering	0,007842	0,016751	0,251934	0,309096
SQL	Notation	0,01408	0,013099	0,294029	0,331961
SQL	Constructed languages	0,119416	0,130216	0,235347	0,26315
SQL	Formalism (deductive)	0	0	0,124249	0,159996
SQL	Abstraction	0,006657	0,009694	0,21549	0,247307
SQL	Scientific disciplines	-1	-1	-1	-1
SQL	Academic disciplines	0	0	0,18708	0,208814
SQL	Applied disciplines	-1	-1	-1	-1
SQL	Social philosophy	0,005142	0,00858	0,238216	0,261346
SQL	Corruption	0	0,006994	0,215465	0,238504
SQL	Core issues in ethics	0,004476	0,004279	0,269043	0,287773
SQL	Society	0	0	0,165758	0,168309
SQL	Economics	0,004781	0,008467	0,226053	0,24357
SQL	Technology	0,02138	0,022321	0,279444	0,30875
Umbral SQL	Branding	0,005629	0,006575	0,199041	0,205378
SQL	Strategic management	0,030945	0,03938	0,233189	0,270035
SQL	Science	0	0	0,222124	0,245381
SQL	Libraries	0,001766	0,009169	0,195172	0,199713
SQL	Technology by type	0	0	0,189636	0,224231
SQL	Electronics	0,006433	0,011471	0,193082	0,215781
SQL	Numbers	0,003836	0,003605	0,134703	0,149089
SQL	Thought	0,005443	0,019116	0,213327	0,261507
SQL	Systems engineering	0,034933	0,048826	0,278478	0,327082
SQL	Conceptual systems	0,012797	0,010951	0,194948	0,231595
SQL	Scientific modeling	0,019018	0,039597	0,28782	0,284886
SQL	Communication	0,018383	0,023538	0,243078	0,273658
SQL	Information systems	0,02126	0,022395	0,300715	0,341412
SQL	Languages	0,145732	0,150193	0,209638	0,230338
SQL	Theories of deduction	0,004215	0,007563	0,174625	0,204312
SQL	Formalism	0	0	0,083942	0,100596
SQL	Innovation	0,007858	0,011344	0,204419	0,23412
SQL	Structure	0,036108	0,032335	0,216176	0,23708
SQL	Creativity	0,003042	0,01039	0,221784	0,23897
SQL	Problem solving	0,001466	0,03041	0,235566	0,274921
SQL	Academia	0,009864	0,015264	0,204015	0,224098
SQL	Knowledge	0,003265	0,006483	0,19322	0,232299

As can be seen in the results the evaluation of the whole content of the page yields more intuitive results. This happens because the whole text contains more context and it is more descriptive. The name of a commercial database for itself do not have any meaning, but if we use the whole text somewhere can be stated that the name in fact is the name for a commercial database.

The use of stemming does not change the results dramatically, an average difference of 0.026 between the stemmed and not stemmed versions. However the process of stemming introduces a lot of processing, and some sort of ambiguity as some words although different share a common lexical root.

With this results in mind, it seems that the best option is the use of complete texts and no stemming.

Now let us see the results obtained when the scope is changed and the comparison is performed with the current category. Categories are compared to their parents, changing the context to the last selected category. The results are presented in Table 3.12.

The behaviour of data is similar to the previous case with some exceptions. In this case, in some occasions, the non stemmed version has higher values than the stemmed one. However these cases are not numerically significant. The average difference on the results according to use of stemming increases a bit till 0.029

The best source for text is the content again, despite of the change of the scope. The best options for text processing seem to be the same as before, use the whole content and avoid the use of stemming.

Data in Table 3.13 show results obtained when changing the scope to perform the comparison of the evaluated category with the aggregate of previously selected categories.

In this case, the results are consistent with the previous case, and again results support the use of the whole text to obtain similarity scores closer to that assigned by a human. Regarding the use of stemming, in this case the means difference has a value of 0.015, this not represents a significant improvement, so we decide to preserve the original terms and not use stemming.

In this method the set of accepted categories provides a context for comparison to new categories. Acceptance of a category depends on the comparison value to be greater than a predefined threshold. Until now we have set the threshold value to 0.

Figure 3.31 shows the results obtained for the cosine measure using the three different comparison scopes. Using this chart we can compare the behaviour of the three approaches. The graph shows relatedness values obtained for each category in a path from the *SQL* category to the root category. The values are presented as a line graph, although values are not continuous, because it eases comprehension and allows to visualize the evolution of comparisons in the path traversed. The X axis presents the first 57 categories as visited in the path to the root. The number of categories used it is very high and normally it is not advisable to select so many categories for a single term, but it is useful for analysis purposes, especially to observe the behaviour when the highest parts of the category graph are visited.

As can be seen, the values obtained for the comparison scope of the start category decrease as the distance between categories increase. Although this behaviour is appropriate the comparison falls to low values very quick. Some times this low values do not really

Table 3.12: Similarity values using cosine similarity and comparing with the current category

C_1	C_2	Title	Title Stem	Content	Content Stem
SQL	SQL	1	1	1	1
SQL	Database management systems	0,304376	0,328664	0,677177	0,707665
SQL	Query languages	0,371164	0,367437	0,827852	0,827352
Database management systems	Databases	0,892666	0,899408	0,769328	0,797359
Query languages	Data management	0,132704	0,138707	0,485707	0,519533
Query languages	Domain-specific progr. lang.	0,596813	0,615075	0,584702	0,640474
Databases	Computer data	0,277491	0,247963	0,629324	0,613666
Databases	Computer science	0,028494	0,042263	0,403903	0,417417
Databases	Data management	0,522408	0,514267	0,821896	0,829481
Data management	Information retrieval	0,088216	0,100013	0,43692	0,465145
Data management	Data	0,803786	0,796463	0,797424	0,755998
Data management	Project management	0,109177	0,127823	0,327099	0,387746
Data management	Computer data	0,782773	0,747379	0,877312	0,860367
Data management	Information technology management	0,293526	0,322526	0,62889	0,662024
Domain-specific progr. lang.	Programming languages	0,497085	0,496621	0,560899	0,642089
Computer data	Computing	0,345238	0,455776	0,505093	0,540648
Computer data	Data	0,697769	0,65835	0,816976	0,78809
Computer science	Formal sciences	0,313014	0,380175	0,58991	0,652623
Computer science	Applied sciences	0,380737	0,3669	0,395112	0,388474
Computer science	Computing	0,331721	0,612425	0,651178	0,769369
Information retrieval	Information science	0,380578	0,398468	0,667745	0,668404
Data	Information	0,116781	0,115412	0,371772	0,426646
Project management	Production and manufacturing	0,339065	0,375296	0,634544	0,660427
Project management	Management	0,633094	0,658671	0,710356	0,740569
Project management	Product development	0,099551	0,110693	0,478313	0,515428
Information technology management	Project management	0,56776	0,585897	0,615111	0,658131
Information technology management	Information technology	0,283871	0,2936	0,593609	0,615588
Programming languages	Programming language topics	0,46291	0,495351	0,557495	0,652799
Programming languages	Computer programming	0,339422	0,353253	0,549872	0,627985
Programming languages	Computer science	0,032774	0,028571	0,282119	0,316411
Programming languages	Computer languages	0,374166	0,377127	0,529434	0,603907
Computing	Digital technology	0,022273	0,017688	0,502949	0,520097
Computing	Information technology	0,00607	0,006812	0,402967	0,425397
Formal sciences	Formalism (deductive)	0,012982	0,255697	0,366737	0,45823
Formal sciences	Interdisciplinary fields	0,126091	0,149776	0,601835	0,63658
Formal sciences	Academic disciplines	0,062257	0,058909	0,551427	0,554904
Formal sciences	Abstraction	0,047749	0,073531	0,625494	0,658125
Formal sciences	Scientific disciplines	-1	-1	-1	-1
Applied sciences	Applied disciplines	-1	-1	-1	-1
Applied sciences	Scientific disciplines	-1	-1	-1	-1
Information science	Information	0,733911	0,756794	0,843111	0,848336
Information science	Science studies	0,229521	0,226224	0,507931	0,55939
Information science	Library and information science	0,37243	0,357558	0,718539	0,70134
Information science	Interdisciplinary fields	0,119425	0,149779	0,524447	0,581429
Information	Concepts	0,034544	0,040109	0,328778	0,372178
Production and manufacturing	Industry	0,334258	0,323528	0,821572	0,831221
Management	Business	0,515309	0,51874	0,807687	0,837719
Management	Accountability	0,00789	0,017303	0,309289	0,34578
Management	Business economics	0,119122	0,138799	0,639037	0,651661
Management	Applied sciences	0,059859	0,06473	0,480441	0,519124
Product development	Production and manufacturing	0,134512	0,168837	0,63408	0,690635
Product development	Product management	0,469521	0,479216	0,733263	0,762032
Programming language topics	Software engineering	0,068126	0,073915	0,343784	0,398164
Programming language topics	Computer languages	0,628293	0,718516	0,779052	0,831248
Computer programming	Software engineering	0,146886	0,138194	0,478342	0,502169
Computer programming	Computing	0,275743	0,372619	0,664065	0,683313
Computer languages	Constructed languages	0,623729	0,677029	0,615072	0,668688
Computer languages	Notation	0,075703	0,071453	0,61608	0,695488
Computer languages	Computing	0,099662	0,138274	0,538598	0,562938
Digital technology	Technology by type	0,055225	0,058076	0,441583	0,501271
Digital technology	Electronics	0,006958	0,099901	0,34649	0,413963
Digital technology	Numbers	0	0	0,155995	0,178852
Formalism (deductive)	Theories of deduction	0,020253	0,018288	0,345585	0,430728
Formalism (deductive)	Formalism	0,736075	0,736075	0,282171	0,332505
Interdisciplinary fields	Science	0,141954	0,16724	0,600879	0,649294
Interdisciplinary fields	Academic disciplines	0,589675	0,584759	0,688798	0,722291
Academic disciplines	Academia	0,254348	0,246497	0,531081	0,545426
Abstraction	Structure	0,127631	0,131168	0,396312	0,41262
Abstraction	Problem solving	0,062192	0,074015	0,522403	0,564591
Abstraction	Innovation	0,004629	0,004383	0,317201	0,327542
Abstraction	Thought	0,096205	0,162478	0,647879	0,694428
Abstraction	Creativity	0,043011	0,058705	0,473352	0,479047
Scientific disciplines	Science	-1	-1	-1	-1
Applied disciplines	Knowledge	-1	-1	-1	-1
Applied disciplines	Academic disciplines	-1	-1	-1	-1
Science studies	Science	0,398527	0,390513	0,798803	0,825923
Science studies	Interdisciplinary fields	0,331764	0,357568	0,685087	0,732098
Library and information science	Libraries	0,40237	0,567685	0,560576	0,637819
Library and information science	Interdisciplinary fields	0,084199	0,101469	0,504981	0,521348
Concepts	Thought	0,146342	0,198449	0,586776	0,63813
Industry	Business	0,066907	0,0722	0,568339	0,591716
Industry	Technology	0,097557	0,095763	0,583478	0,595273
Business	Society	0,015175	0,014951	0,375766	0,351659

Table 3.13: Similarity values using cosine similarity and comparing with the aggregated category. Threshold $t = 0$

C_1	C_2	Title	Title Stem	Content	Content Stem
SQL	SQL	1	1	1	1
Aggregate	Query languages	0,371164	0,367437	0,827852	0,827352
Aggregate	Database management systems	0,327942	0,351925	0,681302	0,706528
Aggregate	Data management	0,437959	0,446218	0,644862	0,677359
Aggregate	Domain-specific progr. lang.	0,189186	0,199383	0,491127	0,519241
Aggregate	Databases	0,691516	0,69215	0,874227	0,88207
Aggregate	Information technology manag.	0,279342	0,295601	0,57816	0,609254
Aggregate	Project management	0,289615	0,310336	0,446148	0,4979
Aggregate	Information retrieval	0,137922	0,15538	0,507571	0,533955
Aggregate	Computer data	0,445687	0,418896	0,70128	0,701084
Aggregate	Data	0,531519	0,51246	0,643685	0,621554
Aggregate	Programming languages	0,099156	0,104455	0,334035	0,372
Aggregate	Computer science	0,095512	0,125232	0,47663	0,496003
Aggregate	Information technology	0,163572	0,162198	0,473734	0,51359
Aggregate	Management	0,506998	0,521892	0,656218	0,698284
Aggregate	Production and manufacturing	0,397228	0,44749	0,739812	0,76375
Aggregate	Product development	0,078611	0,091338	0,559222	0,605722
Aggregate	Information science	0,274618	0,288351	0,680288	0,718743
Aggregate	Computing	0,130903	0,162256	0,661451	0,639176
Aggregate	Information	0,339832	0,384237	0,596389	0,643617
Aggregate	Programming language topics	0,160787	0,201369	0,488255	0,50473
Aggregate	Computer languages	0,238024	0,310756	0,564037	0,601707
Aggregate	Computer programming	0,247213	0,276049	0,641924	0,642651
Aggregate	Formal sciences	0,23799	0,281699	0,686153	0,700282
Aggregate	Applied sciences	0,156195	0,169609	0,570522	0,597699
Aggregate	Accountability	0,068976	0,072004	0,377388	0,387633
Aggregate	Business	0,354924	0,354976	0,671301	0,703344
Aggregate	Business economics	0,126084	0,155757	0,621593	0,636309
Aggregate	Industry	0,120654	0,125547	0,65731	0,658927
Aggregate	Product management	0,381494	0,411413	0,608498	0,641803
Aggregate	Science studies	0,122766	0,134892	0,47168	0,502151
Aggregate	Interdisciplinary fields	0,125209	0,160494	0,557145	0,578208
Aggregate	Library and information science	0,26901	0,277139	0,59249	0,593142
Aggregate	Digital technology	0,084833	0,090456	0,538354	0,592265
Aggregate	Concepts	0,058823	0,066023	0,505413	0,514445
Aggregate	Software engineering	0,164487	0,199809	0,540104	0,607935
Aggregate	Notation	0,050904	0,060588	0,481741	0,501981
Aggregate	Constructed languages	0,167363	0,18824	0,410563	0,403475
Aggregate	Formalism (deductive)	0,003554	0,013602	0,289182	0,33764
Aggregate	Abstraction	0,042029	0,05721	0,512159	0,538632
Aggregate	Scientific disciplines	-1	-1	-1	-1
Aggregate	Academic disciplines	0,158491	0,161503	0,561643	0,563688
Aggregate	Applied disciplines	-1	-1	-1	-1
Aggregate	Social philosophy	0,116056	0,129719	0,628043	0,638044
Aggregate	Corruption	0,039096	0,05427	0,526848	0,540871
Aggregate	Core issues in ethics	0,064109	0,083801	0,695715	0,696117
Aggregate	Society	0,051697	0,048884	0,495446	0,463496
Aggregate	Economics	0,13573	0,144158	0,657753	0,661327
Aggregate	Technology	0,257459	0,263857	0,719346	0,727904
Aggregate	Branding	0,081884	0,077148	0,485751	0,473202
Aggregate	Strategic management	0,436191	0,421189	0,68576	0,708269
Aggregate	Science	0,155356	0,181164	0,6256	0,646649
Aggregate	Libraries	0,077927	0,105587	0,377802	0,367713
Aggregate	Technology by type	0,150487	0,169675	0,558182	0,595147
Aggregate	Electronics	0,135963	0,143378	0,452894	0,462237
Aggregate	Numbers	0,018313	0,022222	0,281001	0,270355
Aggregate	Thought	0,088857	0,148526	0,632018	0,689373
Aggregate	Systems engineering	0,360133	0,391528	0,685933	0,708911
Aggregate	Conceptual systems	0,234271	0,280634	0,555828	0,585224
Aggregate	Scientific modeling	0,256965	0,288032	0,707746	0,649353
Aggregate	Communication	0,214659	0,248165	0,672867	0,686716
Aggregate	Information systems	0,493789	0,513736	0,745956	0,761261
Aggregate	Languages	0,20565	0,209586	0,337866	0,322582
Aggregate	Theories of deduction	0,070368	0,087992	0,464519	0,484816
Aggregate	Formalism	0,024619	0,034903	0,264533	0,286538
Aggregate	Innovation	0,189453	0,216793	0,597244	0,618486
Aggregate	Structure	0,118094	0,110393	0,515412	0,515151
Aggregate	Creativity	0,10001	0,117684	0,632561	0,631829
Aggregate	Problem solving	0,130528	0,174538	0,670811	0,707744
Aggregate	Academia	0,157911	0,170979	0,581411	0,589117
Aggregate	Knowledge	0,095034	0,109096	0,603933	0,652238

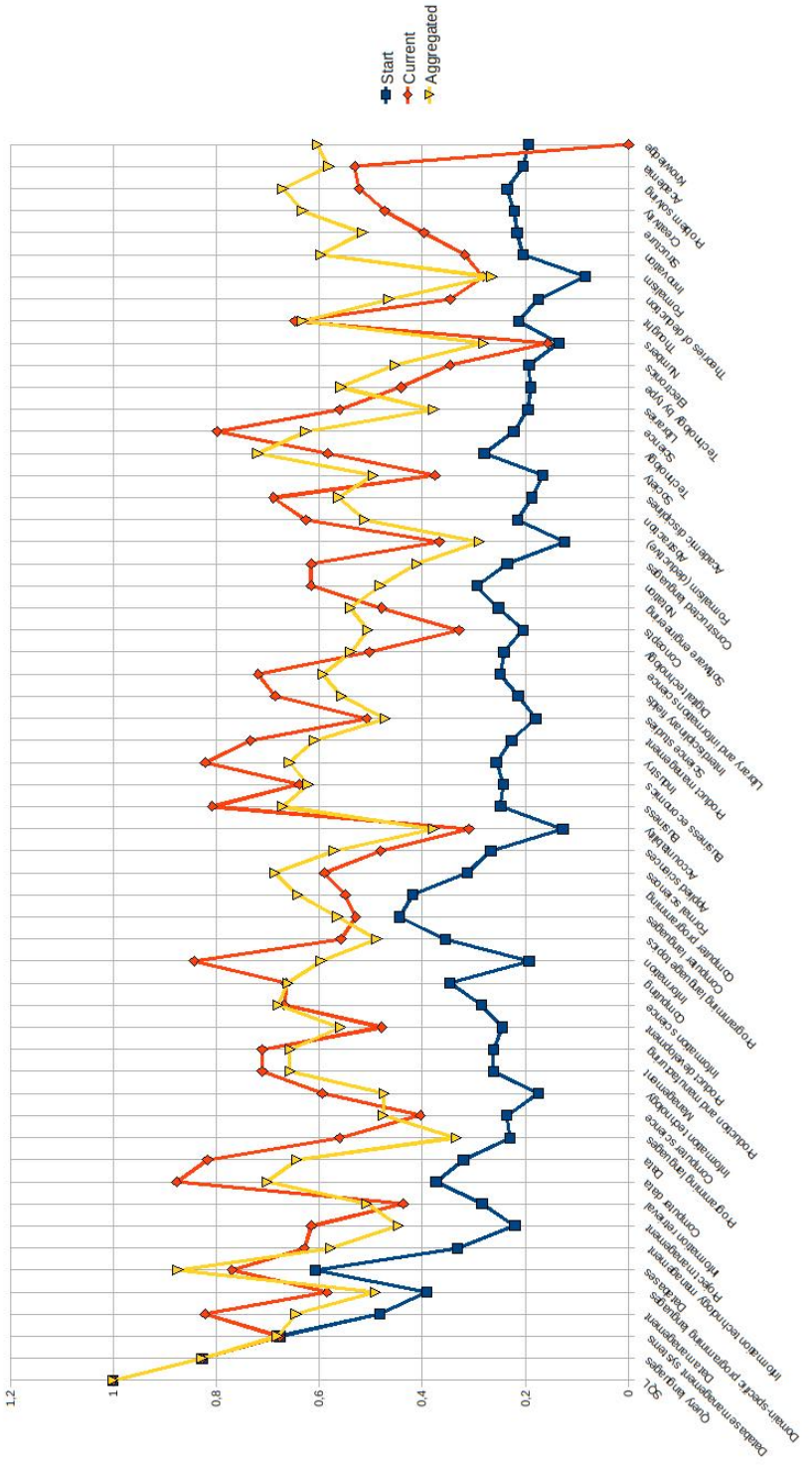


Figure 3.31: Values obtained for the different comparison scopes

reflect the relatedness between the categories. This comparison scope is useful in cases where a local search is desired, but it is ineffective if we want to use a more global context.

When comparing to the current category, the values obtained are very high, as a local search is performed in each step when changing the search context. The continuous update of the context produces the inclusion of new categories by transitivity, a category could be relevant to the current category but not to the start one. This causes the acceptance of many categories, particularly when a general abstract category is accepted.

Finally when using the comparison scope of the aggregated category, we obtain results between the ranges of the previous approaches, but more in line with the latter. In this case the context changes, but is dependent of all the previous accepted categories, obtaining a more global comparison. The context is updated with each new category visited and each time more terms are added. With each new category accepted, the probability to accept to categories increases. This creates a snowball effect that can lead to the acceptance of too many categories.

In the first two methods, the adjustment of the threshold allows to discard non-related categories. However, comparison values do not change with changes in the threshold. In the third method, the comparison is affected by the threshold value, because the threshold determines the accepted categories, and the accepted categories determine the value of the next comparison.

In order to better understand this phenomenon we show the values obtained with the cosine measure and the accumulated category using the whole text of pages and no stemming in Table 3.14. The start category is *SQL*.

We can see how threshold values under 0.4 yield similar values to that obtained for 0.

To better understand the results, Figure 3.32 shows a graphic where relatedness values are computed for thresholds 0, 0.6 y 0.8. Values for threshold 0.2 and 0.4, are omitted because the are the same as for threshold 0.

A value of 0.6 for the threshold yields appropriate values. We must take into account that the lower the threshold, more likely to include noise in the aggregated category. Low values for the threshold can lead to the acceptance of too many categories, and thus to the generation of an ontology too complex.

3.3.9 Evaluation

One criteria to determine the utility of a method against another is the computation of the correlation of its evaluations with human judgements. Semantic relevance of the methods is tested using WordSimilarity-353 [Finkelstein et al., 2002]. WS-353 contains 353 pairs of terms and relatedness judgements provided by humans. Measuring correlation between the relatedness values provided for the pairs by humans and those provided for the methods evaluated, we can determine which makes judgements closer to that provided by humans. As WS-353 contains evaluations over pairs of words and we evaluate pairs of categories, we must adapt the procedure. When determining a category for a term we must determine a page for that term and then select a category from the page. This process introduces some ambiguity as a term can have more than one sense. In order to avoid the noise introduced by

Table 3.14: Relatedness values using cosine measure, complete text and no stemming

C1	C2	t=0.2	t=0.4	t=0.5	t=0.6	t=0.8
SQL	SQL	1	1	1	1	1
Aggregate	Query languages	0,827852	0,827852	0,827852	0,827852	0,827852
Aggregate	Database management systems	0,681302	0,681302	0,681302	0,681302	0,681302
Aggregate	Data management	0,644862	0,644862	0,644862	0,644862	0,505226
Aggregate	Domain-specific progr. lang.	0,491127	0,491127	0,491127	0,491127	0,492952
Aggregate	Databases	0,874227	0,874227	0,877316	0,877316	0,618161
Aggregate	Information technology management	0,57816	0,57816	0,573712	0,573712	0,348162
Aggregate	Project management	0,446148	0,446148	0,443339	0,337646	0,235174
Aggregate	Information retrieval	0,507571	0,507571	0,503546	0,45288	0,342017
Aggregate	Computer data	0,70128	0,70128	0,732226	0,730017	0,398294
Aggregate	Data	0,643685	0,643685	0,67328	0,676484	0,341691
Aggregate	Programming languages	0,334035	0,334035	0,317617	0,316505	0,297461
Aggregate	Computer science	0,47663	0,476293	0,464142	0,420397	0,274925
Aggregate	Information technology	0,473734	0,473855	0,46596	0,334445	0,199953
Aggregate	Management	0,656218	0,656345	0,571243	0,429472	0,279559
Aggregate	Production and manufacturing	0,739812	0,739798	0,70681	0,492438	0,328966
Aggregate	Product development	0,559222	0,559178	0,537218	0,364294	0,270105
Aggregate	Information science	0,680288	0,680311	0,666512	0,526929	0,327811
Aggregate	Computing	0,661451	0,66118	0,653485	0,564401	0,387709
Aggregate	Information	0,596389	0,596494	0,587402	0,394274	0,221543
Aggregate	Programming language topics	0,488255	0,487535	0,47668	0,474777	0,460882
Aggregate	Computer languages	0,564037	0,563459	0,520611	0,522679	0,553718
Aggregate	Computer programming	0,641924	0,641365	0,596103	0,570669	0,488518
Aggregate	Formal sciences	0,686153	0,686118	0,684331	0,594602	0,354772
Aggregate	Applied sciences	0,570522	0,57053	0,566903	0,439905	0,303144
Aggregate	Accountability	0,377388	0,377461	0,376322	0,24507	0,146593
Aggregate	Business	0,671301	0,670825	0,669128	0,39943	0,270366
Aggregate	Business economics	0,621593	0,621477	0,622345	0,395208	0,26741
Aggregate	Industry	0,65731	0,657109	0,656376	0,426892	0,282489
Aggregate	Product management	0,608498	0,608809	0,6066	0,376171	0,246094
Aggregate	Science studies	0,47168	0,471324	0,467862	0,31135	0,210949
Aggregate	Interdisciplinary fields	0,557145	0,5564	0,532747	0,345273	0,255409
Aggregate	Library and information science	0,59249	0,590447	0,584793	0,413647	0,28579
Aggregate	Digital technology	0,538354	0,538121	0,537951	0,413209	0,268123
Aggregate	Concepts	0,505413	0,503952	0,494814	0,338386	0,243458
Aggregate	Software engineering	0,540104	0,540528	0,525315	0,379556	0,27513
Aggregate	Notation	0,481741	0,481835	0,452884	0,368361	0,385808
Aggregate	Constructed languages	0,410563	0,410418	0,380186	0,305618	0,339114
Aggregate	Formalism (deductive)	0,289182	0,289268	0,276465	0,192975	0,143383
Aggregate	Abstraction	0,512159	0,51141	0,485561	0,34959	0,261521
Aggregate	Scientific disciplines	-1	-1	-1	-1	-1
Aggregate	Academic disciplines	0,561643	0,561176	0,545048	0,346249	0,211876
Aggregate	Applied disciplines	-1	-1	-1	-1	-1
Aggregate	Social philosophy	0,628043	0,626403	0,602468	0,38135	0,274228
Aggregate	Corruption	0,526848	0,524998	0,524563	0,369256	0,239994
Aggregate	Core issues in ethics	0,695715	0,694405	0,68571	0,429463	0,307867
Aggregate	Society	0,495446	0,494694	0,492781	0,271146	0,190729
Aggregate	Economics	0,657753	0,657126	0,659287	0,377689	0,253696
Aggregate	Technology	0,719346	0,71965	0,717005	0,476493	0,310554
Aggregate	Branding	0,485751	0,485703	0,489524	0,298298	0,225891
Aggregate	Strategic management	0,68576	0,686278	0,690635	0,3825	0,249799
Aggregate	Science	0,6256	0,625516	0,61336	0,397075	0,253671
Aggregate	Libraries	0,377802	0,37748	0,378249	0,250344	0,234054
Aggregate	Technology by type	0,558182	0,559293	0,563224	0,326102	0,206964
Aggregate	Electronics	0,452894	0,452229	0,448985	0,30129	0,218783
Aggregate	Numbers	0,281001	0,280339	0,275172	0,22455	0,148098
Aggregate	Thought	0,632018	0,63196	0,620816	0,360531	0,248318
Aggregate	Systems engineering	0,685933	0,688806	0,684854	0,471572	0,313051
Aggregate	Conceptual systems	0,555828	0,55628	0,553702	0,352974	0,221465
Aggregate	Scientific modeling	0,707746	0,7089	0,705933	0,516702	0,332176
Aggregate	Communication	0,672867	0,672016	0,666364	0,40488	0,285793
Aggregate	Information systems	0,745956	0,748236	0,758764	0,574107	0,328681
Aggregate	Languages	0,337866	0,336823	0,300936	0,23629	0,323633
Aggregate	Theories of deduction	0,464519	0,461795	0,446315	0,272026	0,216033
Aggregate	Formalism	0,264533	0,263744	0,258391	0,142332	0,096116
Aggregate	Innovation	0,597244	0,599286	0,601217	0,332931	0,225024
Aggregate	Structure	0,515412	0,515123	0,512773	0,387891	0,248661
Aggregate	Creativity	0,632561	0,63123	0,626502	0,366636	0,253686
Aggregate	Problem solving	0,670811	0,670663	0,667011	0,395471	0,268009
Aggregate	Academia	0,581411	0,578411	0,576784	0,32258	0,232297
Aggregate	Knowledge	0,603933	0,603311	0,600913	0,337298	0,222631

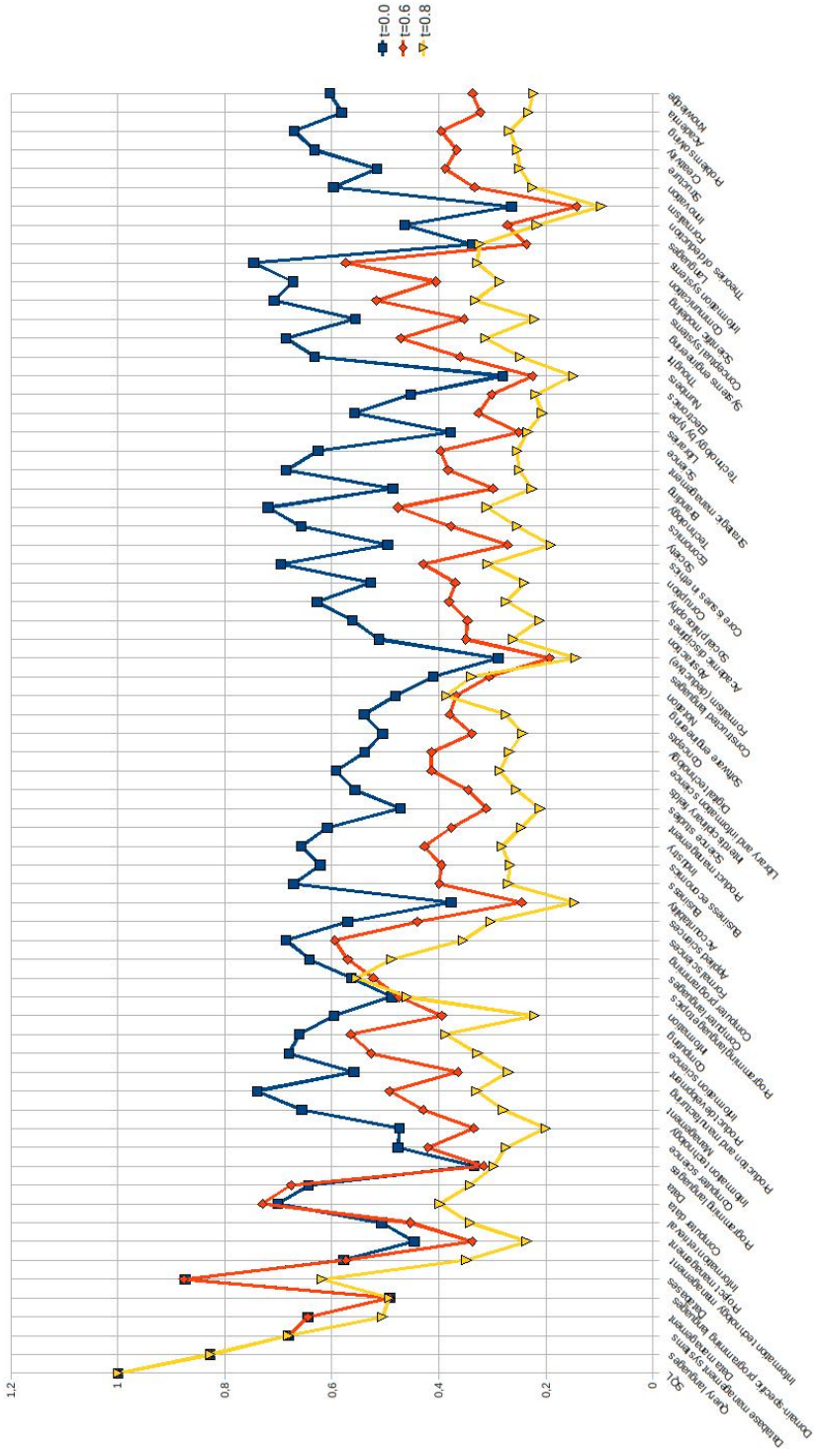


Figure 3.32: Relatedness values using different thresholds

the disambiguation procedure we select an unambiguous subset of WS-353 containing 209 term pairs (WS-209). Another source of ambiguity is the category selection from the page; this is avoided using all categories for the pages and doing pairwise comparison, we take the average value of the comparisons performed and the best value. Evaluation of semantic similarity measures and information content measures is performed using an extension over JWPL API. Correlation values obtained are computed using Pearson correlation coefficient and are presented in Table 3.15. All the tests have been conducted over a dump of the english Wikipedia for August the 17th 2010, stored in a MySQL database.

Table 3.15: Correlation for the Methods

Method	Correlation WS-209
Statistical Indexes	
Average Jaccard	0,2559
Best Jaccard	0,2993
Average Sorensen	0,2741
Best Sorensen	**0,3122**
Average Mountford	0,2303
Best Mountford	0,2841
Semantic Similarity Measures	
Average Rada	-0,2974
Best Rada	-0,2942
Average Leacock-Chodorow	0,3358
Best Leacock-Chodorow	**0,3635**
Average Wu-Palmer	0,2764
Best Wu-Palmer	0,2635
Information Content Measures	
Average Resnik	0,2670
Best Resnik	0,3032
Average Lin	0,2760
Best Lin	**0,3113**
Average Jiang-Conrath	-0,0565
Best Jiang-Conrath	0,1079
Information Retrieval Measures	
Average Cosine	**0,4993***
Best Cosine	0,4176

In order to compare the different approaches, we propose an application to perform concept querying using text terms. The application generates a structured representation of our research interests, which is stored in a database in the form of research article titles. This application generates a conceptual search profile from a set of texts in the form of a taxonomy-like ontology. This conceptual profile will be used to retrieve conceptually similar texts from other sources, although it also can be used to navigate through the original texts.

We extract frequent itemsets from unstructured text and build a taxonomy-like ontology, where each concept in the ontology is annotated with search terms extracted from WordNet. These search terms provide a way to search for a concept over a set of unstructured texts. This ontology can be seen as a summarized version of the texts in the database and as a profile for searching.

For our experiments we compiled four datasets, `CCIA` containing 500 titles of computer science research articles, `MEDLINE` containing 500 titles of biomedicine research articles, `SPORT` containing 500 headlines from Reuters sport news, and `MISC` including 500 randomly selected titles from the previous datasets.

Using a threshold range from $0,6$ to 1 with an increment of $0,025$, we generated ontologies for all measure based methods, and for the four datasets.

To test and evaluate the methods we perform conceptual searches using the ontologies generated. For each concept in the ontology we perform a query over the original datasets using its associated search terms. As the context is the same for all texts, we can compare the different approaches using recall over the original set. Over the ontologies we measure size (number of nodes), recall and generation time. The behaviour of the different methods is very similar for threshold values greater or equal than $0,9$. The most significant difference was found for a threshold value of $0,6$.

We use the ontologies obtained using threshold value $0,6$ to conduct a Friedman test to determine a ranking of the algorithms according to recall. Average ranks obtained by each method in the Friedman test can be seen in Table 3.16.

Algorithm	Ranking
JACCARD	7
SORENSEN	7
MOUNTFORD	7
OVERLAP	7
RADA	7
WU-PALMER	1
RESNIK	7
LIN	7
JIANG-CONRATH	2.25
COSINE	2.75

Table 3.16: Average Rankings of the algorithms (Friedman)

Friedman statistic (distributed according to chi-square with 9 degrees of freedom): 23.618182.

P-value computed by Friedman Test: 0.004948.

Iman and Davenport statistic (distributed acc. to F-distribution with 9 and 27 degrees of freedom): 5.722467.

P-value computed by Iman and Daveport Test: 0.000190465963.

Friedman test determines that Wu-Palmer measure is the best. But let us see if post hoc tests confirm this hypothesis. Adjusted P-values obtained through the application of the post hoc methods (Friedman) are presented in Tables 3.17 and 3.18.

The post hoc tests determine that there are significant differences between Wu-Palmer measure and the rest of algorithms except for Jiang-Conrath and Cosine. We cannot statistically asses the differences between Wu-Palmer, Jiang-Conrath and Cosine, although the ranking clearly declares Wu-Palmer as the best. However, the correlation for Cosine was the best.

i	algorithm	unadjusted p	p_{Bonf}	p_{Holm}	$p_{Hochberg}$	p_{Hommel}
1	JACCARD	0.005069	0.045624	0.045624	0.015208	0.015208
2	SORENSEN	0.005069	0.045624	0.045624	0.015208	0.015208
3	MOUNTFORD	0.005069	0.045624	0.045624	0.015208	0.015208
4	OVERLAP	0.005069	0.045624	0.045624	0.015208	0.015208
5	RADA	0.005069	0.045624	0.045624	0.015208	0.015208
6	RESNIK	0.005069	0.045624	0.045624	0.015208	0.015208
7	LIN	0.005069	0.045624	0.045624	0.015208	0.015208
8	COSINE	0.413686	3.723176	0.827372	0.559305	0.559305
9	JIANG-CONRATH	0.559305	5.033745	0.827372	0.559305	0.559305

Table 3.17: Adjusted p-values (FRIEDMAN) (I)

i	algorithm	unadjusted p	$p_{Holland}$	p_{Rom}	p_{Finner}	p_{Li}
1	JACCARD	0.005069	0.04471	0.015208	0.04471	0.011372
2	SORENSEN	0.005069	0.04471	0.015208	0.04471	0.011372
3	MOUNTFORD	0.005069	0.04471	0.015208	0.04471	0.011372
4	OVERLAP	0.005069	0.04471	0.015208	0.04471	0.011372
5	RADA	0.005069	0.04471	0.015208	0.04471	0.011372
6	RESNIK	0.005069	0.04471	0.015208	0.04471	0.011372
7	LIN	0.005069	0.04471	0.015208	0.04471	0.011372
8	COSINE	0.413686	0.656236	0.559305	0.451538	0.484194
9	JIANG-CONRATH	0.559305	0.656236	0.559305	0.559305	0.559305

Table 3.18: Adjusted p-values (FRIEDMAN) (II)

Wu-Palmer method has the main disadvantage of having the highest generation time from the three best ranked methods. The computation of Wu-Palmer and Jiang-Conrath relies in the calculation of the LCS, as we explained previously this implies the adaptation of Wikipedia category graph to a hierarchical structure. This adaptation breaks some relations from the graph and thus some semantics are lost. The Cosine measure do not relies in the computation of a hierarchy from the graph, but performs a lot of processing in order to compute term vectors.

The selection of the method to use in each case may vary as different applications have different requirements and thus our selection may change according to the specifics of each problem. If a quick generation time is needed we can discard Wu-Palmer, if the amount of memory available is low, we discourage the use of Cosine. The main disadvantage of Jiang-Conrath is the use of the hierarchical translation of the graph. This structure is computed once, so if we are going to generate many ontologies using this method, computation of the hierarchy is only done once.

A more detailed compilation of results and statistical tests is performed in the main document of the Thesis.

3.4 Conclusions

In this chapter we have presented a proposal for a general methodology to extract semantics from unstructured texts using text mining techniques and external knowledge resources. The methodology is comprised of several stages, syntactic preprocessing, semantic preprocessing, intermediate representation and semantic extension. Each stage of this methodology can be implement using a wide variety of tools. In this chapter we propose and analyze two implementations based on WordNet and Wikipedia.

The WordNet implementation of the methodology uses text mining techniques to obtain an intermediate representation based on AP-Sets. The AP-Sets have been transformed into a basic taxonomy-like ontology with the aid of WordNet lexical database. Expressing texts as ontologies, even as simple ones, allows to describe textual attributes and perform semantic queries about the content of text. We talk about semantic queries because these are based on concepts from ontology instead of query terms. In the end the realization of a query is a set of terms, but in this case those terms are determined by the concept to search for and not the other way around.

The use of a lexicographic dictionary as WordNet allows to obtain meaning for the text term to process. Obtaining the meaning of all words in a text seems not very useful as the generated ontology can be too general and too big. In order to summarize the most important information from texts, the AP-Sets intermediate representation is used. This way, the most relevant concepts are identified based on their frequency and the ontology generated is more compact and represents the relevant parts of the domain.

Semantic preprocessing is a task particularly appropriate to perform in conjunction with WordNet. Semantic preprocessing is the task of finding the correct senses for terms and group all the terms with the same sense. In this regard WordNet is a useful tool as part of disambiguation algorithms, and as a reference to create synonym groups.

As we have seen, using WordNet for semantic extension presents several drawbacks. WordNet is designed by linguist for linguistic task, and thus the taxonomy is biased towards a linguistic perspective, while we seek a more general real-world representation of concepts. Moreover WordNet lacks of domain specific technical vocabulary for certain topics.

Trying to overcome WordNet limitations in the task of semantic extension we have explored the possibilities of using Wikipedia. The previous stages of the methodology are very well suited to the WordNet structure so we maintain them as in the WordNet implementation.

The collaborative effort in Wikipedia has a clear and immediate effect, topic coverage of Wikipedia is better than that of WordNet. In addition the structure is grounded in real world concepts using less general abstract concepts than WordNet. Wikipedia contains a graph-like structure, the Wikipedia category graph. This graph can be used as a reference taxonomy in the semantic extension stage of the methodology. In order to use properly Wikipedia category graph, we have studied different measures to determine semantic relatedness between categories (nodes in the wikipedia category graph).

We have analyzed various approaches for Wikipedia graph exploration. Terms in the intermediate representation given by AP-Sets are mapped to Wikipedia categories. Using a graph exploration method together with a relatedness measure, it is possible to traverse

a path in the graph while selecting categories which are similar to the one given. We have analyzed the results provided by each of the classical measures for relatedness used.

Using different measures we have generated ontologies for the texts. We propose an automatic evaluation of the ontologies obtained based on its use in an application. The application proposed is the creation of a ontology from a text attribute in a database. This ontology must serve as a summary of the content of the text attribute and as a means to generate queries based on the concepts present in the ontology. The ontology is built taking information in the intermediate representation and mapping it to Wikipedia category graph. Representative sub-graphs of Wikipedia are selected for each term and are used to create the ontology conceptual hierarchy. Each concept in the ontology is annotated with synonym terms to extend queries. Queries are created for each concept using its synonyms and all of the concepts subsumed by it. Queries are tested against the original data to asses the coverage of the ontology built.

Measuring the number of nodes obtained, recall and the time necessary to process the ontology, we select the most appropriate method for different particular situations.

Future work will include the storage of the new ontologies inside the database using the schema presented in Chapter 2. We plan on creating more complex ontologies adding additional relations extracted from Wikipedia apart from the hierarchical ones, based on links between pages. Other external semantic sources such as DBPedia, ConceptNet or Snomed can be used in addition to Wikipedia and WordNet to detect complex relations.

We plan on implementing a tool for semantic querying of database texts using the generated ontologies as a Protégé plug-in.

Chapter 4

Conclusions

We have presented a proposal to deal with information semantics in relational databases. In our proposal we decided to use a object-relational database to take advantage of its extension capabilities.

Throughout this work we have approached the study of information semantics at two levels:

- **Providing semantics to database structures.**

  For structure semantics we use knowledge representation by means of ontologies.

  - Domain OWL ontologies are used to describe the conceptual level of a database schema. We have presented a hybrid schema able to deal with ontology and instances storage, without information loss. Storing the ontology in the database along with its instances allows to perform semantic queries on data.
  - The use of a domain ontology as a tool for designing database schemas, eases the process of developing database schemas for Semantic Web data. The storage of the ontology inside the database allows to preserve the semantics of the schema generated from it. Knowing the model from which the schema has been generated, allows its processing and can be used as documentation for the schema.

- **Providing semantics to database content.**

  Management of content semantics is treated at attribute level. We distinguish between management of numeric attributes and text attributes.

  - *Numeric attributes.* The only way to improve semantics of a numeric attribute is through the use of more meaningful queries. Using fuzzy logic in databases, numeric attributes can store uncertainty. The use of uncertainty in the attributes and in the queries, allows the user to express flexible queries. These queries can ask for numeric values expressed as approximate values, ranges, possibility distributions, etc. This way of expressing queries is closer to the way a human agent would express them.

- *Text attributes.* We have proposed a methodology to extract semantics from non-structured texts. This methodology is general enough to be implemented using a wide variety of tools. In our case we have studied different implementation options and tools for each stage in the methodology. Particularly, we have centered our attention towards the representation and semantic extension stages. For text representation we have used text mining techniques to build an AP-Set based representation. From this intermediate representation we have created basic taxonomy-like ontologies.

  The extension of AP-Sets has been performed using external knowledge sources.

  * *Semantic extension using WordNet.* We have implemented the methodology using WordNet. We tested the use of WordNet in two stages of the methodology, semantic preprocessing and semantic extension. Semantic preprocessing using WordNet allows to group synonym terms and determine a representative of the set. This representative is used to avoid the use of multiple terms to designate the same concept. WordNet is a tool perfectly suited for the task of synonym detection. However, the use of WordNet for semantic extension gives poor results due to the linguistic bias and low coverage of WordNet.
  * *Semantic extension using Wikipedia.* The coverage, continuous update and concept graph of Wikipedia, offered enough evidence to perform a deep study. We have studied and evaluated different similarity and relatedness measures on Wikipedia category graph in order to generate relevant ontologies for the texts processed.

    For the evaluation of the ontologies obtained using each different approach, we have searched in literature for works on the matter. From the three possibilities which we considered best suited for our case, expert evaluation and the comparison to a gold standard cannot be performed. Due to the high number of ontologies obtained the evaluation by an expert was not possible. The lack of reference ontologies for each possible text set, made this approach not very useful. The only approach that was adaptable to our case and could be generalized to different scenarios was the use of the ontology in the context of an application.

    The application proposed for ontology evaluation is based on concept queries, and the coverage of this concepts over the original texts. The studies conducted determine that categories content and the position in the graph determines relatedness between categories better than other factors.

Future work will continue the different research directions opened.

- **Ontology representation in relational databases.**

  We plan on developing elemental reasoning inside the database, the adaptation to OWL2 language and dealing with ontology evolution inside our schema. An interesting future research direction is the merging and mapping of schemas created using our approach.

- **Semantic extension of unstructured texts**.

  Future work is oriented to ontological relations extraction using WordNet, Wikipedia and other external knowledge resources such as DBPedia or ConceptNet. We plan the development of a Protégé plug-in for query answering and schema visualization for databases using our approach. Through these Protégé plug-ins we intend to bring the possibility to manage the ontologies stored in the database using traditional ontology management frameworks. At first, Protégé will only be used to ontology creation and visualization, as editing is conditioned to advances achieved in ontology evolution in our schema.

Storage and management of the ontologies generated from texts inside the database, will complete the cycle of our proposal. This way the whole ontology management can be performed inside the database.

Bibliography

[OMD, 2005] (2005). Ontology definition metamodel. Third Revised Submission to OMG/ RFP ad/2003-03-40.

[Pro, 2008] (2008). The Protégé ontology editor and knowledge acquisition system. http://protege.stanford.edu/.

[Agirre and Edmonds, 2007] Agirre, E. and Edmonds, P., editors (2007). *Word Sense Disambiguation: Algorithms and Applications*, volume 33 of *Text, Speech and Language Technology*. Springer.

[Agrawal and R.Srikant, 1994] Agrawal, R. and R.Srikant (1994). Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, pages 487–499.

[Andreasen and Bulskov, 2009] Andreasen, T. and Bulskov, H. (2009). Conceptual querying through ontologies. *Fuzzy Sets and Systems*, 160(15):2159 – 2172. Special Issue: The Application of Fuzzy Logic and Soft Computing in Information Management.

[Astrova et al., 2007] Astrova, I., Korda, N., and Kalja, A. (2007). Storing owl ontologies in sql relational databases. *International Journal of Electrical, Computer, and Systems Engineering*, 1(4):242–247.

[Baader and Werner, 2003] Baader, F. and Werner, N. (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*, chapter Basic Description Logics, pages 47–100. Cambridge University Press, New York, NY, USA.

[Bagui, 2009] Bagui, S. (2009). Mapping OWL to the Entity Relationship and Extended Entity Relationship models. *International Journal of Knowledge and Web Intelligence*, 1(1):125–149.

[Bagui and Earp, 2003] Bagui, S. and Earp, R. (2003). *Database design using entity-relationship diagrams*. Auerbach Publications.

[Banerjee and Pedersen, 2002] Banerjee, S. and Pedersen, T. (2002). An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet. In *Proceedings of the Conference on Computational Linguistics and Intelligent Text Processing (CICLING)*, pages 136–145. Springer.

[Barranco et al., 2008a] Barranco, C. D., Campaña, J. R., and Medina, J. M. (2008a). A b+-tree based indexing technique for fuzzy numerical data. *Fuzzy Sets and Systems*, 159(12):1431–1449. Advances in Intelligent Databases and Information Systems.

[Barranco et al., 2005a] Barranco, C. D., Campaña, J. R., Cubero, J. C., and Medina, J. M. (2005a). A fuzzy object relational approach to flexible real estate trade. *WSEAS Transactions on Information Science and Applications*, 2(2):155–160.

[Barranco et al., 2005b] Barranco, C. D., Campaña, J. R., and Medina, J. M. (2005b). Improving query expressiveness in product search interfaces using fuzzy logic. *WSEAS Transactions on Business and Economics*, 2(2):80–87.

[Barranco et al., 2008b] Barranco, C. D., Campaña, J. R., and Medina, J. M. (2008b). *Handbook of Research on Fuzzy Information Processing in Databases*, chapter Towards a Fuzzy Object-Relational Database Model, pages 431–461. Hershey, PA, USA, first edition.

[Barranco et al., 2007] Barranco, C. D., Campaña, J. R., Medina, J. M., and Pons, O. (23-26 July 2007). On storing ontologies including fuzzy datatypes in relational databases. *Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International*, pages 1–6.

[Barrasa et al., 2004] Barrasa, J., Corcho, O., and Gómez-Pérez, A. (2004). R2O, an extensible and semantically based database-to-ontology mapping language. In *Proceedings of the Second Workshop on Semantic Web and Databases (SWDB2004)*.

[Batini et al., 1992] Batini, C., Ceri, S., and Navathe, S. B. (1992). *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin/Cummings.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5):28–37.

[Bizer and Seaborne, 2004] Bizer, C. and Seaborne, A. (2004). D2RQ-Treating Non-RDF Databases as Virtual RDF Graphs. In *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*.

[Borgida et al., 2003] Borgida, A., Lenzerini, M., and Rosati, R. (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*, chapter Description Logics for Databases, pages 462–484. Cambridge University Press, New York, NY, USA.

[Brants, 2000] Brants, T. (2000). TnT: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231.

[Calvanese et al., 1998] Calvanese, D., Lenzerini, M., and Nardi, D. (1998). *Description logics for conceptual data modeling*, chapter 8, pages 229–264. Kluwer Academic Publisher.

[Campaña et al., 2009] Campaña, J. R., Martín-Bautista, M. J., Medina, J. M., and Vila, M. A. (2009). Semantic Enrichment of Database Textual Attributes. *Flexible Query Answering Systems, FQAS 2009*, pages 488–499.

[Chen, 1976] Chen, P. (1976). The Entity-Relationship Model – Toward a Unified View of Data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36.

[Cimiano, 2006] Cimiano, P. (2006). *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*, chapter Ontology Learning from Text, pages 19–34. Springer.

[Cubero et al., 2004] Cubero, J. C., Marín, N., Medina, J. M., Pons, O., and Vila, M. A. (2004). Fuzzy object management in an object-relational framework. In *X Intl. Conf. of information processing and management of uncertainty in knowledge-based systems*, pages 1767–1774.

[Cunningham, 2002] Cunningham, H. (2002). GATE, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254.

[Das et al., 2004] Das, S., Chong, E. I., Eadon, G., and Srinivasan, J. (2004). Supporting ontology-based semantic matching in RDBMS. In *Proceedings of the* 30^{th} *VLDB Conference, Toronto, Canada*, pages 1054–1065.

[Date, 1990] Date, C. (1990). *An Introduction to Database Systems.* Addison-Wesley, Reading, MA.

[Dice, 1945] Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302.

[Fellbaum, 1998] Fellbaum, C. (1998). *WordNet: an electronic lexical database.* MIT Press, Cambridge, MA.

[Finkelstein et al., 2002] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2002). Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

[Gabrilovich and Markovitch, 2007] Gabrilovich, E. and Markovitch, S. (2007). Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1606–1611.

[Giménez and Márquez, 2004] Giménez, J. and Márquez, L. (2004). Svmtool: A general pos tagger generator based on support vector machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 43–46.

[Jaccard, 1901] Jaccard, P. (1901). Etude comparative de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Société vaudoise des Sciences Naturelles*, (37):547–579.

[Jean et al., 2006] Jean, S., Pierra, G., and Ait-Ameur (2006). Domain ontologies: a database-oriented analysis. In *Web Information Systems and Technologies (WEBIST 2006)*, pages 238–254.

[Jiang and Conrath, 1997] Jiang, J. and Conrath, D. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the 10th international conference on research in computational linguistics, Taipei, Taiwan*, pages 19–33.

[Kalyanpur et al., 2006] Kalyanpur, A., Parsia, B., Sirin, E., Grau, B. C., and Hendler, J. (2006). Swoop: A web ontology editing browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(2):144–153.

[Leacock and Chodorow, 1998] Leacock, C. and Chodorow, M. (1998). *Combining Local Context and WordNet Similarity for Word Sense Identification*, chapter 11, pages 265–283. The MIT Press.

[Lesk, 1986] Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the ACM-SIGDOC Conference, Toronto, Canada*, pages 24–26. ACM New York, NY, USA.

[Lin, 1998] Lin, D. (1998). An information-theoretic definition of similarity. In Shavlik, J. W., editor, *Proc. 15th International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann.

[Madche and Staab, 2000] Madche, A. and Staab, S. (2000). Discovering Conceptual Relations from Text. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI)*, pages 321–325.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

[Marcus et al., 1993] Marcus, M., Marcinkiewicz, M., and Santorini, B. (1993). Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

[Marín et al., 2006] Marín, N., Martín-Bautista, M. J., Prados, M., and Vila, M. A. (2006). Enhancing Short Text Retrieval in Databases. In *Proceedings of FQAS 2006, Milan, Italy. Lecture Notes in Artificial Intelligence.*, volume 4027, pages 613–624. Springer.

[Martín-Bautista et al., 2008] Martín-Bautista, M. J., Martínez-Folgoso, S., and Vila, M. (2008). A New Semantic Representation for Short Texts. In *Data Warehousing and Knowledge Discovery: 10th International Conference, Dawak 2008 Turin, Italy, September 1-5, 2008*, pages 347–356. Springer.

[Martín-Bautista et al., 2006] Martín-Bautista, M. J., Prados, M., Vila, M. A., and Martinez-Folgoso, S. (2006). A knowledge representation for short texts based on frequent itemsets. In *Proceedings of IPMU 2006, Paris, France.*, pages 1065–1070.

[Martínez-Folgoso, 2008] Martínez-Folgoso, S. (2008). Tratamiento semántico de atributos textuales en un modelo relacional orientado a objetos: Implementación en software libre. Ph.D. Thesis.

[Medina et al., 2002] Medina, J. M., Galindo, J., Berzal, F., and Serrano, J. M. (2002). Using object relational features to build a fuzzy database server. In *VIII Intl. Conf. of information processing and management of uncertainty in knowledge-based systems (IPMU 2002)*, pages 307–314.

[Medina et al., 1994] Medina, J. M., Pons, O., and Vila, M. A. (1994). GEFRED: A generalized model of fuzzy relational databases. *Information Sciences*, 76(1-2):87–109.

[Mehler, 2009] Mehler, A. (2009). A quantitative graph model of social ontologies by example of Wikipedia. *Genres on the Web: Computational Models and Empirical Studies*, pages 291–352.

[Milne and Witten, 2008] Milne, D. and Witten, I. (2008). An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, pages 25–30.

[Mountford, 1962] Mountford, M. D. (1962). *An index of similarity and its application to classification problems*, pages 43–50. P.W.Murphy editors.

[Navathe and Elmasri, 2007] Navathe, S. and Elmasri, R. (2007). *Fundamentals of Database Systems*. Addison Wesley.

[Patwardhan et al., 2003] Patwardhan, S., Banerjee, S., and Pedersen, T. (2003). Using measures of semantic relatedness for word sense disambiguation. In Gelbukh, A., editor, *Computational Linguistics and Intelligent Text Processing (CICLing 2003)*, volume 2588 of *Lecture Notes in Computer Science*, pages 241–257. Springer Berlin / Heidelberg.

[Pokorný and Vojtáš, 2001] Pokorný, J. and Vojtáš, P. (2001). A data model for flexible querying. *In Proc. ADBIS'01, Lecture Notes in Computer Science*, 2151:280–293.

[Porter, 1980] Porter, M. (1980). An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137.

[Prade and Testemale, 1984] Prade, H. and Testemale, C. (1984). Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. *Information Sciences*, 34:115–143.

[Rada et al., 1989] Rada, R., Mili, H., Bicknell, E., and Blettner, M. (1989). Development and application of a metric on semantic nets. *IEEE Transactions on Systems Management and Cybernetics*, 19(1):17–30.

[Resnik, 1995] Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *In Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453.

[Resnik, 1999] Resnik, P. (1999). Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of artificial intelligence research*, 11(95):130.

[Roldán García et al., 2005] Roldán García, M., Navas Delgado, I., and Aldana Montes, J. (2005). A design methodology for semantic web database-based systems. In *ICITA (1)*, pages 233–237.

[Schmid, 1999] Schmid, H. (1999). Improvements in part-of-speech tagging with an application to German. *Natural language processing using very large corpora*, 11:13–26.

[Seco et al., 2004] Seco, N., Veale, T., and Hayes, J. (2004). An intrinsic information content metric for semantic similarity in WordNet. In *Proceedings of ECAI-04*, volume 16, pages 1089–1090.

[Sorensen, 1948] Sorensen, T. (1948). A method of establishing groups of equal amplitude in plant sociology based on similarity ofspecies content and its application to analyses of the vegetation on danish commons. *Dan. Vidensk. Selsk. Biol. Skr.*, (5):1–34.

[Straccia, 2006] Straccia, U. (2006). A fuzzy description logic for the semantic web. In Sanchez, E., editor, *Fuzzy Logic and the Semantic Web*, Capturing Intelligence, chapter 4, pages 73–90. Elsevier.

[Strube and Ponzetto, 2006] Strube, M. and Ponzetto, S. (2006). WikiRelate! Computing semantic relatedness using Wikipedia. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, pages 1419–1424.

[Sugumaran and Storey, 2006] Sugumaran, V. and Storey, V. C. (2006). The role of domain ontologies in database design: An ontology management and conceptual modeling environment. *ACM Trans. Database Syst.*, 31(3):1064–1094.

[Teorey, 1989] Teorey, T. J. (1989). Distributed database design: A practical approach and example. *SIGMOD Record*, 18(4):23–39.

[Thalheim, 1993] Thalheim, B. (1993). Foundations of entity - relationship modeling. *Ann. Math. Artif. Intell.*, 7(1-4):197–256.

[Todirascu et al., 2001] Todirascu, A., de Beuvron, F., Galea, D., and Rousselot, F. (2001). Using description logics for ontology extraction. In *Proc. of ROMAND'2000 Workshop on Robust Parsing*, pages 89–105.

[Torsten Zesch, 2007] Torsten Zesch, I. G. (2007). Analysis of the wikipedia category graph for nlp applications. pages 1–8.

[Toutanova et al., 2003] Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259.

[Toutanova and Manning, 2000] Toutanova, K. and Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pages 63–70.

[Trinkunas and Vasilecas, 2007] Trinkunas, J. and Vasilecas, O. (2007). A graph oriented model for ontology transformation into conceptual data model. *INFORMATION TECHNOLOGY AND CONTROL*, 36(1A):126–132.

[Umano, 1982] Umano, M. (1982). Freedom-O: A fuzzy database system. *Fuzzy Information and Decision Processes*.

[Vasilescu et al., 2004] Vasilescu, F., Langlais, P., and Lapalme, G. (2004). Evaluating variants of the Lesk approach for disambiguating words. In *Proceedings of the Conference of Language Resources and Evaluations (LREC 2004)*, pages 633–636.

[Vysniauskas and Nemuraite, 2006] Vysniauskas, E. and Nemuraite, L. (2006). Transforming ontology representation from OWL to relational database. *Information Technology And Control*, 35A(3):333–343.

[Woelk and Tomlinson, 1994] Woelk, D. and Tomlinson, C. (1994). The Infosleuth project: Intelligent Search Management via Semantic Agents. In *Second World Wide Web Conference '94:Mosaic and the Web*.

[Wu and Palmer, 1994] Wu, Z. and Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138, Morristown, NJ, USA. Association for Computational Linguistics.

[Zemankova-Leech and Kandel, 1985] Zemankova-Leech, M. and Kandel, A. (1985). Implementing imprecision in information systems. *Information Sciences*, 37:107–141.

[Zesch and Gurevych, 2010] Zesch, T. and Gurevych, I. (2010). Wisdom of crowds versus wisdom of linguists–measuring the semantic relatedness of words. *Natural Language Engineering*, 16(01):25–59.

[Zobel and Moffat, 1998] Zobel, J. and Moffat, A. (1998). Exploring the similarity space. In *ACM SIGIR Forum*, volume 32, pages 18–34.