UNIVERSITY OF GRANADA

Departament of Computer Science and
Artificial Intelligence

PhD Program:
Design, Analysis and Applications of Intelligent Systems

PhD Thesis Dissertation:

DOCUMENT CLASSIFICATION MODELS BASED ON BAYESIAN NETWORKS

PhD Student:
Alfonso Eduardo Romero López

Advisors:
Prof. Dr. Luis M. de Campos Ibáñez
and Prof. Dr. Juan M. Fernández-Luna

Editor: Editorial de la Universidad de Granada
Autor: Alfonso Eduardo Romero López
D.L.: GR 2884-2010
ISBN: 978-84-693-2527-8

UNIVERSITY OF GRANADA

Departament of Computer Science and
Artificial Intelligence

PhD Program:
Design, Analysis and Applications of Intelligent Systems

PhD Thesis Dissertation:

MODELOS DE CLASIFICACIÓN DOCUMENTAL BASADOS EN REDES BAYESIANAS

PhD Student:
Alfonso Eduardo Romero López

Advisors:
Prof. Dr. Luis M. de Campos Ibáñez
and Prof. Dr. Juan M. Fernández-Luna

Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain.

Alan M. Turing, *Computing Machinery and Intelligence* (1950)

A mis padres y a mi hermano

To my parents and my brother

Acknowledgements

This work was carried out at the Department of Computer Science and Artificial Intelligence of University of Granada during the years 2005–2010. The completion of my PhD would not have been made possible without the help of many, who I would like to acknowledge with these words.

I should begin thanking my supervisors: first of all, Prof. Dr. Luis M. de Campos. This work would have not been successful without his advice, knowledge and deep experience in Bayesian networks. Even though text categorization might not have been his domain of expertise, I feel fortunate he work with me on it, while defining a new research line. Secondly, Prof. Dr. Juan M. Fernández-Luna, who gave the help necessary to develop this work, from an Information Retrieval approach. Thus, being supervised by this tandem, has been a pleasure for me, and I must say I have learnt a lot from both from the scientific and the personal dimension.

It also was a real pleasure to work with Prof. Dr. Juan F. Huete, who was always very supportive, and with my colleagues in the group UTAI, Miguel A. Rueda and Carlos Martín-Dancausa. I also would like to thank Dr. Andrés Masegosa his help in developing a classifier combining Weka and Elvira, which lead to the development of the last work supporting this thesis. I also must thank him for the LaTeX format and layout for this memory.

I have a special mention for the people from the people of the LIP6, at the University Pierre et Marie Curie (Paris, France), who accepted my internship as a PhD student from September to December 2008. In spite of the very short length of the period, I must say I really had

there a great time, both scientifically and personally. I thank Prof. Dr. Patrick Gallinari for his support, and for his scientific proposal on collective classification, which opened new horizons in my work. I thank Prof. Dr. Ludovic Denoyer too, his interest on me, and his advice for developing solutions to some problems. Indeed, he is one of the best Java hackers I have ever met. I also thank him his consent for being a member of the jury of this PhD. Besides, I would like to thank, with no particular order, to all people I met in the lab, which made my internship a lot of easier to carry out: Séverine Guillaume, Stéphane Peters, Jean-François Pessiot, Francis Maes, Anna Ciesielska, Vihn Truong, Sheng Gao, Marc-Ismaël Akodjenou, Alex Spengler. Rudy Sicard and very specially David Buffoni.

Other people who deserve my most sincere acknowledge are some of my colleagues at the department, who are or were working as funded PhD students. Thank you all for the good moments, and for all your personal support: Fernando Bobillo, Carlos Cano, Fernando García, Ignacio García, Juan Ramón González, Eduardo Eisman, Aída Jiménez, Javier López, Víctor López, Antonio Masegosa, Miguel Molina, Mariló Ruiz, Julián Garrido and, of course, Alberto Fernández. I also would like to thank Pablo García (for his friendship and for the great after-lunch coffees), and Prof. Dr. JJ Merelo from the Department of Computer Architecture (for his support in some bad moments).

More in the personal point, I have walked this long way enjoying the company of my friends. Without their presence, my life would not be so full. Thank you to the *mistery-men team*: Bruno, Chema, Elena, Inés, Jesús, José Enrique, Salva, Manu, Marina, Pepe, and Tere; the *Brotherhood team*: María, David, Ana, Javi C., José Carlos, Lupe, Paqui, Juan Antonio, Benito, Fisco and Lala; the *Bocket team*: Óscar, Javier, Víctor, Bernar, Miguel E., Álex, Alejandro and Jorge; and of course, Antonio M., as the only survivor of the *Van Gogh team*. Thank you all for proposing a *Druid's tonight*, for all the enriching

Fridays at 9 pm, for sharing the *Paladium Therapy* with me, for your *que no te pase ná* and for always having time for a beer with me at untimely hours.

Prof. Dr. José Almira, from the University of Jaén, needs a mention apart. I would like to thank him his deep and great friendship, and for giving me the opportunity to publish with him, even before starting this PhD.

In the last place, but not less important, I give my heartfelt thanks to my both parents Alfonso and Ana, and my brother Diego. Thank you for your company and love, for your example, for inspiring me every day of my life, and for always trusting in me.

This doctoral thesis has been supported by the Spanish Ministerio de Ciencia e Innovación with the FPU scholarship AP2005-0617 and the project TIN2008-06566-C04-01, the project of the Spanish research programme Consolider Ingenio 2010: MIPRCV (CSD2007-00018), and the projects of the Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía TIC-04526 and TIC-276.

Agradecimientos

Este trabajo ha sido desarrollado en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada entre los años 2005–2010. Su finalización no habría sido posible sin la ayuda de muchos, a quien me gustaría dedicar estas palabras.

Querría comenzar agradeciendo a mis directores: lo primero de todo, al Prof. Dr. Luis M. de Campos. Este trabajo no habría podido realizarse sin sus consejos, conocimientos y su amplia experiencia en las redes bayesianas. Incluso aunque la clasificación documental podría no ser su campo de trabajo, me siento afortunado porque haya trabajado conmigo, mientras definíamos una nueva línea de investigación. En segundo lugar, al Prof. Dr. Juan M. Fernández-Luna, quien ha proveído la ayuda necesaria para realizar este trabajo, desde el punto de vista de la Recuperación de Información. Por tanto, ha sido un placer para mí ser dirigido por este tándem y debo añadir que he aprendido bastante de ambos, desde la dimensión científica y personal.

Ha sido también un placer trabajar con el Prof. Dr. Juan F. Huete, que siempre me ha apoyado mucho, y con mis colegas en el grupo UTAI, Miguel A. Rueda y Carlos Martín-Dancausa. También quiero agradecer al Dr. Andrés Masegosa su ayuda para desarrollar un clasificador combinando Weka y Elvira, que condujo al desarrollo de la última publicación contenida en esta tesis. También debo agradecerle el formato LaTeX de esta memoria.

Tengo una mención especial para la gente del LIP6, de la Universidad Pierre et Marie Curie (París, Francia), que aceptaron mi estancia como estudiante de doctorado de septiembre a diciembre de 2008. A

pesar de la poca duración de ese período, debo decir que realmente pasé allí una estancia maravillosa, tanto científica como personalmente. Le agradezco al Prof. Dr. Patrick Gallinari su aceptación y apoyo, y también su propuesta científica en clasificación colectiva, que abrió nuevos horizontes en mi trabajo. Quiero agradecer también al Prof. Dr. Ludovic Denoyer, su interés en mí, y sus consejos a la hora de realizar soluciones a algunos problemas. Sin duda alguna, él es uno de los mejores hackers de Java que jamás he conocido. También quiero agradecerle su consentimiento para ser miembro del tribunal de esta tesis. Además, también me gustaría agradecer, sin un orden en particular, a toda la gente que encontré en el laboratorio, que hicieron mi estancia mucho más fácil de sobrellevar: Séverine Guillaume, Stéphane Peters, Jean-François Pessiot, Francis Maes, Anna Ciesielska, Vihn Truong, Sheng Gao, Marc-Ismaël Akodjenou, Alex Spengler, Rudy Sicard, y muy especialmente David Buffoni.

Otra gente que merece mi más sincero agradecimiento son algunos de mis colegas del departamento, que están o estuvieron trabajando como becarios. Gracias a todos por los buenos momentos y por vuestro apoyo personal: Fernando Bobillo, Carlos Cano, Fernando García, Ignacio García, Juan Ramón González, Eduardo Eisman, Aída Jiménez, Javier López, Víctor López, Antonio Masegosa, Miguel Molina, Mariló Ruiz, Julián Garrido y, sobre todo, Alberto Fernández. También me gustaría dar gracias a Pablo García (por su amistad y por los cafés de sobremesa, entre otras cosas), y al Prof. Dr. JJ Merelo (por su apoyo en algunos momentos difíciles), del departamento de Arquitectura y Tecnología de Computadores.

Más en el plano personal, he recorrido este camino disfrutando de la compañía de mis amigos. Sin su presencia, mi vida ahora no sería tan plena. Gracias al *equipo mistery-men*: Bruno, Chema, Elena, Inés, Jesús, José Enrique, Salva, Manu, Marina, Pepe, y Tere; el *equipo fraternidad*: María, David, Ana, Javi C., José Carlos, Lupe, Paqui, Juan Antonio, Benito, Fisco y Lala; el *equipo Bocket*: Óscar, Javier,

Víctor, Bernar, Miguel E., Álex, Alejandro y Jorge; y sobre todo, a Antonio M., como único superviviente del *equipo Van Gogh*. Gracias a todos por proponer un *Druid's tonight*, por esos enriquecedores viernes a las 9, por compartir conmigo la *Terapia Paladium*, por vuestros *que no te pase ná* y por tener siempre tiempo para echar una cerveza conmigo a horas intempestivas.

Mención aparte merece el Prof. Dr. José Almira de la Universidad de Jaén. Quiero agradecerle aquí su gran y sincera amistad, y el darme la oportunidad de publicar con él, incluso antes de comenzar este trabajo.

En el último lugar pero no menos importante, quiero dar mi más profundo agradecimiento a mis padres Alfonso y Ana, y a mi hermano Diego. Gracias por vuestra compañía y vuestro cariño, por vuestro ejemplo, por inspirarme cada día de mi vida, y por siempre confiar en mí.

Esta tesis doctoral ha sido financiada por el Ministerio de Ciencia e Innovación con la beca FPU AP2005-0617 y el proyecto TIN2008-06566-C04-01, por el proyecto del programa de investigación Consolider Ingenio 2010: MIPRCV (CSD2007-00018), y los proyectos de la Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía TIC-04526 y TIC-276.

Contents

List of Figures

List of Tables

Part I

Introduction and Foundations

Introduction

Motivation

With the growth of the Internet and the huge spread of the digital computer in the 1990s and the 2000s, a very high amount of information, mainly composed of electronic Text Documents, have been made available in an exponential manner. In order to give easier access to electronic information, and reduce that "information overload" several solutions have been proposed in the literature. One example of these solutions are Information Retrieval systems which are able to return from a collection the set of documents that matches some user needs, captured by means of a query. Another solution are Text Categorization systems, aimed to present the information organized in a set of topics or categories. They are designed to automatically label the documents with categories (corresponding to certain generic topics, often with a very defined semantic meaning), which can be previously learnt by the system with a set of preclassified examples in this topic, making the navigation easier in the collection. In this work we shall study this second problem, that is to say, how documents can be automatically organized in a set of classes.

In fact, the field of *Supervised Document Categorization* [119] (often called *Automated Indexing*, *Text Filtering* or *Text Routing*) is active since 1961 [83], and can be roughly defined as the task of labeling documents with the categories of a predefined set, using a function learnt (the classifier) with examples already classified. For obvious reasons, this problem falls between the domains of Machine Learning (mainly for the techniques used to build the classifiers) and Information Retrieval (which provided the tools for document processing and their treatment by computers) areas. This is probably why it has attracted –mainly since 1995–

lot of researchers of both communities with great interest in this kind of problems, producing a notably amount of publications [118].

The applications of the developments made on this area are fairly diverse. Perhaps the figurehead of them is spam email detection [114]. The problem of discriminating between real and junk email is present on every email account, and the benefits of using such a categorization system are translated into a huge amount of time and money saved everywhere. This task is a problem of binary Text Categorization (that is to say, the set of categories has size equal to two: "spam" and "not spam") where even the simplest models, as the Naïve Bayes [85] have obtained good results.

Another important application is Automatic Indexing of official or scientific documents [73]. In this case, the documents should be organized in a set of hundreds or thousands of categories, a task that is made manually in many scenarios. Besides, instead of having a flat set of labels, sometimes they are identified with the descriptors of a thesaurus [21], which adds some metadata and large hierarchy. In contrast with the previous case, here we can assign an arbitrary subset of labels to each document, and the set of categories is notably larger than a simple dichotomy. The problem of classifying documents in a hierarchy of classes is very typical of this area and implies using models which are more elaborate than the classical Machine Learning ones [96].

Beyond the categorization of just flat documents, a trending topic in Text Categorization in the last years is Structured Document Categorization. Here we use "structured" for both XML categorization [9] (the document is not atomic, but composed of different structural units), and link-based Document Categorization (where we have a structure of explicit relationships among the documents) [79]. The last methods have found direct applications as solutions to the graph labeling (labeling a set of linked documents using link information in addition to just the textual content) and the webspam detection problems [25] (detecting group of pages whose content is spam, often linked among them to confuse the user and appear on the first places of the results given by a web search engine).

The Bayesian networks [102] framework was chosen as a very appropriate tool to provide possible solutions in this dissertation. These probabilistic models have shown great success in presenting interesting solutions to both problems

of Machine Learning (concretely classifiers) [1] and Information Retrieval [15]. Moreover, the Naïve Bayes classifier [85] (and, in general, many of the probabilistic classifier models) can be studied using this framework. Thus, we have used this formalism to benefit from all the general research done in these models [99].

Main Contributions of the Dissertation

The first contribution of the dissertation is to give several new Text Categorization methods based on noisy OR gates [102] as a discriminative counterpart of the multinomial Naïve Bayes classifier. The Naïve Bayes classifier is widely used in the Machine Learning and the Text Categorization communities, and represents a good starting point to work with probabilistic models. In order to overcome several limitations of the approach, we also provide an *ad-hoc* pruning procedure that refines the learning process of our OR gate model. We claim that the proposed OR gate models maintain the simplicity of the Naïve Bayes approach, increasing its discrimination power.

The second contribution of the dissertation is the introduction of the thesaurus-based indexing problem. This problem has been previously treated on the literature, but either as a supervised categorization problem (with no use of the hierarchy or the metadata) or as an unsupervised indexing problem. We shall present a formalization of a thesaurus, independent of the classification model described afterwards, and suitable for many of the most commonly used thesauri. Together with this formalization, we shall state the problem of thesaurus based categorization, and we shall propose two solutions, one using training information, and other with no use of it, both built on a Bayesian network-based model of the thesaurus and its related information. In fact, the model with training information is shown as an extension of the unsupervised one, making use of the previously presented OR gate-based classification model. We shall try to show that a probabilistic model of the relationships of the categories and the metadata of the thesaurus, together with training information, can provide a categorization power comparable or superior to the state-of-the art in Supervised Text Categorization (the Linear Support Vector Machine [60]).

Our contribution ends with a proposal of several models for the Structured Text Categorization problem. Firstly we shall make some XML document transformations in order to reduce to flat documents and apply the noisy OR gate models. On the other hand, we shall show two solutions for the link-based document categorization problem; one for the multiclass case (where a document is labeled with one among several categories), and other for the multilabel one (where the number of associated categories is free). Both proposed models are based on a Bayesian network learnt directly from the relationships among the categories, present on the training data, and making use of a probabilistic classifier for the content (like, for instance, the Naïve Bayes). In this way, the models can also be seen, as an extension of a classic probabilistic model for the case of the link-based classification.

Chapter Overview

This dissertation is arranged into three parts. The first one, Part I, is an Introduction to the main results, providing a preface (this introduction), and two chapters with the foundations needed to understand the content. Concretely, the chapter 1 provides a brief introduction to the supervised Text Categorization problem, presenting the main problem, describing several models with detail, and explaining how to evaluate different solutions. In order to complete the foundations part, chapter 2 introduces the basic concepts of probability theory used here, and those from the Bayesian networks language, as graphical separation, learning algorithms or inference algorithms. In this last case, one learning and and one inference algorithms are presented because they will be used later.

Part II contains the main contributions of this dissertation, presented on previous section. Thus, in chapter 3 we describe the OR gate classifier, together with its pruning procedure. In chapter 4 we deal with the thesaurus-based classification problem explained before. Finally, in chapter 5 both the structured and the link-based document classification problems are discussed, along with our suggested solutions.

Finally, Part III contains the last chapter of this dissertation, where the conclusions and the future lines of work are stated, as well as we review the list of publications supporting the contributions of this thesis.

Introducción

Motivación

Con el crecimiento de la Internet y el gran éxito de los ordenadores en los 90 y principios de los 2000, ha aparecido, de forma exponencial, una gran cantidad de información, compuesta fundamentalmente de documentos textuales. Para dar un acceso más fácil a la información electrónica, y reducir la "sobrecarga de información" se han propuesto varias soluciones en la literatura. Un ejemplo de estas soluciones son los sistemas de Recuperación de Información, capaces de devolver documentos de una colección, que sean relevantes a las necesidades de un usuario, formuladas con una consulta. Otra solución son los sistemas de Clasificación Documental, dirigidos a presentar la información organizada en un conjunto de clases o categorías. Estos sistemas se diseñan para etiquetar automticamente a los documentos con categoras (correspondientes a temas genérico con un significado semántico muy definido), que puede ser aprendido por el sistema con un conjunto de ejemplos preclasificados, haciendo la navegacin por la coleccin ms fcil. En este trabajo estudiaremos este segundo problema, esto es, cómo se pueden organizar automáticamente un conjunto de documentos en una lista de categorías.

De hecho, el campo de la *Clasificación Documental Supervisada* [119] (también llamado algunas veces *Indexación Automática*, *Filtrado Textual* o *Text Routing*) está activo desde 1961 [83], y puede ser definido, a grandes rasgos, como el proceso automático de etiquetado de un conjunto de documentos con las categorías de una lista predefinida, utilizando una función aprendida con ejemplos ya clasificados. Por razones obvias, este problema se encuentra entre los dominios del Aprendizaje Automático (debido fundamentalmente a las técnicas usadas para la

construcción de clasificadores, heredadas de aquél) y la Recuperación de Información (que provee las herramientas para el procesado automático de documentos y su tratamiento algorítmico). Probablemente por esto este campo ha atraído –fundamentalmente desde 1995– gran cantidad de investigadores de ambas comunidades con bastante interés en este tipo de problemas, produciendo una notable lista de publicaciones [118].

La aplicación de lo desarrollado en este área son bastante diversas. Tal vez el mascarón de proa de las mismas es la detección de correo basura (spam) [114]. El problema de discriminar entre correo real y basura se encuentra en toda cuenta de correo, y los beneficios de utilizar un sistema de clasificación para ello se traducen en una enorme cantidad de tiempo y dinero ahorrado en todas partes. Esta tarea es un problema de Clasificación Documental binaria (esto es, el conjunto de categorías tiene tamaño igual a dos: "spam" y "no spam") donde incluso los modelos más simples, como el Naïve Bayes [85] han obtenido buenos resultados.

Otra aplicación importante es la Indexación Automática de documentos oficiales o científicos [73]. En este caso, los documentos deben de organizarse en conjuntos de cientos o miles de categorías, teniendo que ser esta tarea realizada de forma manual en muchos escenarios. Además, en vez de tener una lista normal de clases, a veces éstas se identifican con los *descriptores* de un tesauro [21], lo que añade unos ciertos metadatos además de una estructura de jerarquía. Al contrario que en el caso anterior, aquí podemos asignar un subconjunto de etiquetas de tamaño arbitrario a cada documento, y el conjunto de categorías es notablemente mayor que una simple dicotomía. El problema de clasificar documentos en una jerarquía de clases es muy típico de este área e implica el uso de modelos que son más elaborados que los modelos clásicos de Aprendizaje Automático [96].

Más allá que la clasificación de documentos planos, un tema de actualidad en Clasificación Documental en los últimos años es el de la Clasificación de Documentos Estructurados. Aquí usamos "estructurados" tanto para la clasificación de documentos XML [9] (donde el documento no es atómico, sino que puede estar organizado internamente alrededor de una estructura bien definida), como para la Clasificación Documental basada en enlaces (en la que tenemos una estructura con relaciones explícitas entre los documentos) [79]. Estos últimos métodos tienen aplicación directa a los problemas de etiquetado de grafos (etiquetar un

conjunto de documentos enlazados usando la estructura de enlaces además de sólo usar el texto) y de detección de webspam [25] (detectar grupos de páginas cuyo contenido es spam, en ocasiones enlazadas entre ellas para confundir al usuario y para aparecer en los primeros puestos de los resultados de un buscador web).

Se eligió el formalismo de las Redes Bayesianas [102] para desarrollar todas las soluciones propuestas en esta tesis. Estos modelos probabilísticos han tenido gran éxito al resolver tanto problemas de Aprendizaje Automático (en especial los de clasificación) como de Recuperación de Información [15]. Además, el clasificador Naïve Bayes [85] (y, en general, muchos de los modelos probabilísticos de clasificación) se pueden estudiar usando este marco. Por tanto, se usa este formalismo para beneficiarse de toda la investigación general realizada previamente en estos modelos [99].

Principales Contribuciones de esta Memoria

La primera contribución de esta tesis es presentar nuevos métodos de Clasificación Documental basados en puertas OR ruidosas [102] como una contrapartida discriminativa al clasificador Naïve Bayes multinomial. El clasificador Naïve Bayes se usa bastante en las comunidades de Aprendizaje Automático y en la de Clasificación Documental, y representa un buen punto inicial para trabajar con modelos probabilísticos. Para mejorar algunas limitaciones del modelo, también presentamos un procedimiento de poda *ad hoc* que refina el proceso de aprendizaje de nuestro modelo de puerta OR. Afirmamos que el modelo de puerta OR propuesto mantiene la simplicidad del Naïve Bayes, incrementando su poder de discriminación.

La segunda contribución de esta tesis es la introducción del problema de indexación basada en un tesauro. Este problema se ha tratado anteriormente en la literatura, pero o bien como un problema de clasificación supervisada (sin usar la jerarquía o los metadatos), o como un problema de indexación no supervisada. Presentaremos una formalización de un tesauro, independiente del modelo de clasificación que se describe posteriormente, y apropiado para muchos de los tesauros usados en el mundo. Junto a esta formalización, presentaremos el problema de clasificación en tesauros propiamente dicho, y propondremos dos

soluciones: una usando información de entrenamiento y otra sin usarla, ambas construidas usando un modelo de red bayesiana del tesauro y de su información relacionada. De hecho, el modelo con información de entrenamiento se muestra como una extensión del no supervisado, haciendo uso del clasificador puerta OR anteriormente presentado. Trataremos de probar que un modelo probabilístico de las relaciones entre las categorías y los metadatos que tiene el tesauro, junto con la información de entrenamiento, puede tener un poder de clasificación comparable o superior al modelo que representa el estado del arte en Clasificación Documental (la Máquina de Vectores Soporte Lineal [60]).

Nuestra contribución finaliza con la proposición de varios modelos para problemas de clasificación estructurada. Primeramente realizaremos transformaciones a documentos XML para convertirlos en texto plano y poder aplicar el clasificador puerta OR presentado. Por otra parte, mostraremos dos soluciones al problema de clasificación basada en enlaces; uno para el caso multiclase (donde un documento se etiqueta con una de entre varias categorías) y otro para el modelo multietiqueta (donde el número de categorías asociado a cada documento es libre). Ambas propuestas se basan en redes bayesianas aprendidas directamente de las relaciones entre las categorías presentes en los datos de entrenamiento, y hacen uso de un clasificador probabilístico para el contenido (como, por ejemplo, el Naïve Bayes). De este modo, nuestros modelos pueden ser vistos como una extensión de un modelo probabilístico clásico para el caso de clasificación basada en enlaces.

Visión General de los Capítulos

Esta memoria se divide en tres partes. La primera, Parte I, es una Introducción a los resultados, compuesta por un prólogo (esta introducción), y dos capítulos conteniendo los fundamentos necesario para comprender el resto de contenidos. Concretamente, el capítulo 1 provee una breve introducción a la Clasificación Documental Supervisada, presentando el problema principal, describiendo varios modelos con detalle, y explicando cómo evaluar diferentes soluciones. Para completar la parte de fundamentos, el capítulo 2 introduce los conceptos básicos de Teoría de la Probabilidad usados aquí, y algunos de redes bayesianas como

separación gráfica, algoritmos de aprendizaje o de inferencia. En ese último caso, presentamos en detalle un algoritmo de aprendizaje que será usado más tarde.

La Parte II contiene las contribuciones principales de esta memoria, presentadas en la sección previa. Así, en el capítulo 3 describimos el clasificador puerta OR, junto con su procedimiento de poda. En el capítulo 4 se trata el problema de clasificación basada en tesauros explicado anteriormente. Finalmente, en el capítulo 5 se tratan tanto el problema de la clasificación estructurada, como el de la clasificación basada en enlaces, junto con nuestras soluciones aportadas.

Finalmente, la Parte III contiene el último capítulo de esta memoria, conteniendo las conclusiones y las líneas futuras de trabajo, además de la revisión de la lista de publicaciones que apoyan las contribuciones de esta tesis.

Chapter 1

Text Document Categorization

1.1 Introduction

The task of *Automated Text Categorization* [119] (also known as *Text Classification*) is the process of assigning predefined categories to text documents. This is a very important field in Computer Science, with strong relationships with other areas like Artificial Intelligence, Machine Learning and Information Retrieval. In fact, the number of publications in this area has grown notably since the 1960s, with a huge peak at the beginning of the 2000s, giving more than 500 references in all years (see figure 1.1, extracted from [118], for more details about the number of publications in this area per year).

In this dissertation we shall put our interest on algorithmic methods for Text Categorization (that is to say, those that could be run out by a computer). This is why, from beyond, we shall not be using the "Automated" qualifying (or any of its derivatives), because it is assumed on this context.

Text Categorization algorithms are widely present on many current applications. For instance, email spam detection [114] (where each email can be labeled with "spam" or "not spam"), assigning a set of predefined keywords (labels) to a scientific paper, organizing news stories on a predefined set (national, international, sports, ...), etc. In fact, in almost every environment with a huge number of text documents, where one user can search by a set of predefined subjects, this kind of methods are indeed necessary.

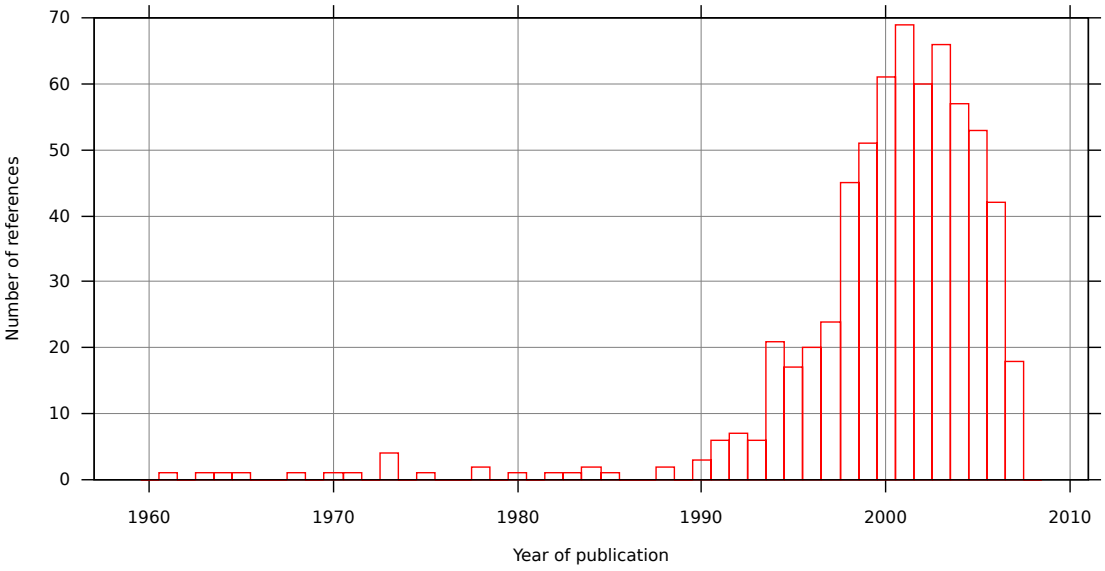

Figure 1.1: Number of publications in Automated Text Categorization, per year. Only publications until 2008 are listed.

This chapter proposes a general view of all the concepts, definitions, and some of the more relevant works on Text Categorization, as a basic knowledge to understand this dissertation, as well as to contextualize it. Therefore, we shall organize the chapter as follows: in section 1.2 we shall review the main definitions concerning this problem, along with some general conventions which will make easier this task. In section 1.3, we shall review some approaches to the building of text classifiers, mainly inspired by Machine Learning techniques.

The peculiarities and difficulties of this problem will be presented on section 1.4, where we shall explain why this is not the same problem as the classic Machine Learning one.

Having different categorization models is not very useful if there are not standard procedures to compare those approaches. In this way, section 1.5 will study the problem of the evaluation of this task (i.e. how well a classifier performs), and section 1.6 will review some testing corpora which are made publicly available in order to make a standard benchmark set, where researchers can test their own algorithms.

1.2 Main Concepts, Definitions and Notation in Text Categorization

In this section we define the different entities that take part in the Text Categorization process. Some of them are concepts which are already known, and therefore we shall explain what we understand by those terms. Besides, we shall explain here the notation which is followed on this entire dissertation.

1.2.1 Documents, Corpora and Categories

A (text) **document** is a succession of words together with some punctuation that form a text. Note that we identify a document with its textual content, and not with the physical document. Moreover, we do not identify different parts in the document and this is why we shall refer to this kind of document as "flat document" (for the case of documents with a internal structure, we shall explain what a Structured Document is in chapter 5).

A **corpus** is another name for "document collection". We shall use both terms indistinctly.

A **document representation** is a typification of a text document, in a format which is easier to understand by computers. We shall present the usual document representations on next subsection.

A **category** or a **class** is a word or a set of words, often associated to a concept (i.e. it has a semantic meaning), which can be used to label documents, according to their contents.

1.2.2 Document Preprocessing

The action of **document preprocessing** is a set of initial document transformations which are useful to manage a document in subsequent stages of Text Categorization. This preprocessing, inherited by that proposed in the Information Retrieval field [108], often includes the following procedures, applied in this order:

1. **Case folding**: all the text in the document is set to lower case.

2. **Removal of punctuation marks**: because they are not going to help in the process of Text Categorization.

3. **Removal of stopword:** this process consists of the removal of words which do not have any useful meaning by themselves (e. g. articles, prepositions, pronouns). Those words are called *stopwords*, and there are standard lists of them for every language. Concretely, in the English language, the 571 stopwords list included in the SMART Retrieval system [117] is the most used one.

4. **Stemming:** a process for reducing inflected (or derived) words to their stem, base or root form. This process is performed automatically by an algorithm, being the Porter's one [105] the most used approach. A term which has been stemmed does not necessarily produces a meaningful word as its output. For simplicity, we shall refer to the stemmed word as a *term*.

Document preprocessing can include, as its final step, the arrangement of all the remaining terms in lexicographical order. Therefore, the position of each term in the document is ignored, and the document is considered as a set of preprocessed terms, with no particular order. This is called the **bag of words** model, where a document is represented as an unordered collection of terms.

This model is similar to the "first-order word approximation" used by Shannon in [123], and reduces a document to a list of unrelated terms usually losing, as a consequence, the contextual meaning and the structure of some expressions present on the text.

Note that preprocessing a document and converting it to the *bag of words* model is not a one-to-one transformation (that is to say, it is impossible to recover the original document from its bag of words form). This is not a problem, but it implies that the form of a preprocessed document is very different than the original one. This fact is also useful if one researcher want to distribute a corpus, but he does not want to give access to the original documents. This is the case of the Reuters RCV1 collection, composed of news articles of the Reuters news agency, distributed by Lewis [78] after its preprocessing.

1.2.3 Representing Documents with Vectors

The usual representation of a document after its preprocessing is the **vector representation**. This representation is very simple, and consists of identifying each term of the document collection with a dimension in a real vector space. Thus, every document becomes a real vector, with a real number as a coordinate, meaning the weight or the importance of the term in the vector.

There is not a unique formula to compute the value of a coordinate for a certain vector, but it is generally agreed that a coordinate of a term is equal to zero if this term does not belong to the document.

We reproduce here a generalization of the formula for the weight of the term i in document j, shown in [7]:

$$w_{ij} = l_{ij}\ g_i\ n_j.$$

In the formula, l_{ij} is a *local* weight of the i-th term in the j-th document. g_i is a *global* weight, a value which is computed once for each term (i-th term in this case). Finally, n_j is a factor of *normalization* which depends only on the current document (the j-th one).

The simplest representations of a document are these two:

- **Binary representation:** a document is a vector with values in $\{0, 1\}$, where the i-th coordinate is equal to 1 if the i-th term appears in d, and 0 otherwise.

- **Frequential representation:** a document is a vector with values in $\mathbb{N} \cup \{0\}$, where the i-th coordinate is equal to the frequency of the i-th term on that document (and 0 if the term does not belong to the document).

In both examples, it is clear that $g_i = 1$ and $n_j = 1$, and the variable part is the l_{ij} value.

Another typical representation, which is useful, for example, for Support Vector Machines or Logistic Regression classifiers [50], follows the so-called *tf-idf* scheme. In it, the l_{ij} is set to a function of the frequency of a term in the document (called the *tf*), and the g_i (the *idf*) is a function of the *inverse document*

frequency of a term. This tf-idf is a technique that has several definitions [116], and has been used extensively in Information Retrieval with good results. The traditional scheme of this representation is to use the frequency of the term in the document as the *tf*, and setting the *idf* of the i-th term to:

$$idf_i = \log\left(\frac{M}{n(t_i)}\right) ,$$

where M stands for the number of documents in the collection, and $n(t_i)$ for the number of documents which contain term t_i. This scheme gives a higher weight w_{ij} to rare terms in the collection, lowering it for common terms (note that the *idf* grows for rare terms, which are terms that occur in few documents).

Finally, in some cases, a *tf-idf* vector is *normalized*. This is generally done by dividing the vector by its Euclidean or l^2-norm (that is to say, n_j gets the inverse of the norm value), obtaining a unit vector (the l^2-normalization). Other normalization schemes are also possible, as the l^1-normalization.

1.2.4 The Supervised Text Categorization Problem

The **problem of text categorization** consists of, roughly speaking, deciding which categories to assign to those documents whose labels are unknown. This problem can be unsupervised or supervised, being this last one the aim of our focus. In this subsection, we shall state the problem without using any formula or special notation, just describing the task.

Building a classifier is a task which consists of *finding an automatic method* (classifier) that, given a new unlabeled document, is capable to assign it only one or several labels. In order to achieve this task, the classifier is provided with a *training set*, which is composed of previously labeled documents. This training set is the only information that can be used to build the classifier.

If only one label is preassigned to the documents in the training corpus, and then, only one label can be assigned to any new documents to classify, we call this problem a *multiclass problem*. A multiclass problem is called a *binary problem* for the specific case that the size of the set of labels is equal to two. If we are in the case that any number of labels can be assigned to a document, this will be called

a *multilabel problem*. Analogously we could also say *multilabel/multiclass/binary corpus*.

In order to test the effectiveness of a classifier inferred from training data, we are also provided of a *test set*. Documents belonging to the test set should be classified with the inferred model, in order to compare their original labels to those obtained by the classifier with an *evaluation measure*. This is a procedure that is useful to compare several approaches to build classifiers on different datasets.

Finally, we shall introduce two sub-modalities on this problem: *hard categorization* and *soft categorization*. Doing *hard categorization* means finding a classifier which is capable of assigning one (or more, if needed) label to a document. On the other hand, *soft categorization* is the problem of finding a method that can give a numeric real value for each pair composed of one category and one document. The soft categorization is a more general approach to build classifiers. In fact, as we shall see on next section, it is very easy to build a hard categorization method from a soft categorization one.

1.2.5 Notation

We shall note as $\mathcal{D} = \{d_1, \ldots, d_m\}$ a document collection. Thus, d_i will be a single document. Observe that, for our purposes, a document and its representation will be the same entity. This is because we shall be always using the same kind of representation for each model, unless specified the opposite. Therefore, $\mathcal{D}$ will be either a set of text documents or more often, a set of vectors representing documents.

The set of categories will be noted $\mathcal{C} = \{c_1, \ldots, c_p\}$, where c_j will be a certain category. The set of terms, on the other hand, will be $\mathcal{T} = \{t_1, \ldots, t_q\}$. With abuse of notation, we shall often write $d_i \in c_k$ to express that a certain document d_i is labeled with category c_k. We shall also use the notation $t_i \in d_j$ to express the fact that the term t_i occurs on the document d_j.

A labeled corpus will be a set $\mathcal{D}$ such that, $\forall d_i \in \mathcal{D}, \exists c_j \in \mathcal{C} : d_i \in c_j$. This is the general case (multilabel corpus), where more than one category can be assigned to a document. If $\mathcal{D}$ verifies that $\forall d_i \in \mathcal{D}, \exists! c_j \in \mathcal{C} : d_i \in c_j$, the corpus is a multiclass one (only one label is assigned to every document). A

multiclass corpus where $|\mathcal{C}| = 2$ is called a binary labeled corpus. When we refer to a "training corpus" it is assumed that we are dealing with a labeled corpus. A "test corpus" will be a corpus whose labels are unknown during the categorization process, but they are available for evaluation purposes.

We can now redefine some of the previously proposed problems, using the presented notation. The problem of **supervised classification**: given a training (labeled) set $\mathcal{D}_{Tr}$, a set of categories $\mathcal{C}$, and a test set $\mathcal{D}_{Te}$, consists of building the mapping

$$f : \mathcal{D} \longrightarrow \mathcal{C},$$

for the binary and the multiclass case, and

$$f : \mathcal{D} \longrightarrow 2^{\mathcal{C}} \setminus \emptyset,$$

for the multilabel one, where $\mathcal{D}$ is the set of all possible documents. The function f should be built using only information available on $\mathcal{D}_{Tr}$ and its labeling, and its quality can be measured comparing the labeling assigned to the documents in $\mathcal{D}_{Te}$, with the real labels, using a evaluation measure (like one of the proposed on section 1.5).

Due to its complexity, the problem of multilabel categorization is often expressed as finding n different binary $f_i, i = 1, \ldots, n$ capable of assigning or not each document to the i-th category (understanding that if a document is assigned a negative label $\overline{c}_i$, it means that it is not labeled with that category):

$$f_i : \mathcal{D} \longrightarrow \{c_i, \overline{c}_i\},$$

Thus, the multilabel problem for n categories takes the form of n binary classification problems.

The previous approaches are examples of the statement of the problem as a *hard categorization* one. That is to say, a classifier is defined as capable of assigning (or not) a label to every document, but all the assignments result similar (there is not a measure of the strength of that assignment).

We can redefine the problem of supervised document categorization, in term of *soft categorization*. A classifier will be then a function g, defined as follows:

$$g : \mathcal{D} \times \mathcal{C} \longrightarrow \mathbb{R}$$

In this case, $g(d_j, c_i)$ is a real value, called the CSV (Categorization Status Value), which measures the strength[1] of the assignment.

Obviously, the treatment of the multilabel problem is similar here, being decomposed on different and independent binary problems. Note that g represents the n classifiers, and this is why we do not need to write g_i.

A way to obtain a hard categorization classifier f from a soft one g is to estimate from training data a real value τ, called a threshold, and defining the new f classifier as follows (we show a binary case for brevity):

$$f(d) = \begin{cases} c & \text{if } g(d, c) \geq \tau \\ \overline{c} & \text{if } g(d, c) < \tau \end{cases}$$

There are several ways that the parameter τ can be set to a certain value. In all of them, training data is often used in several partitions to find the threshold that maximize an evaluation measure.

1.3 Approaches to Building Text Classifiers

1.3.1 Some Important Classifiers

We present here three classic approaches to Text Categorization. The *Multinomial Naïve Bayes* –characterized for being fast and simple– the *Support Vector Machine* classifier (in its *linear version*) –which is the state-of-the-art in Text Categorization–, and the *Rocchio* method, which is highly intuitive and it is used quite a lot. In certain occasions, some of these models will be used on the following chapters as a baseline (a *comparison*).

[1] This "strength" is an intrinsic characteristic of the classifier, and the only requirement is that it is greater if the classifier "trusts" more in this assignment.

1.3.1.1 The Multinomial Naïve Bayes

We should firstly clarify that, in the context of text classification, there exist two well-known different models called Naïve Bayes, the multivariate Bernoulli Naïve Bayes model [67; 72; 109] and the multinomial Naïve Bayes model [75; 85]. In this section we shall only consider the multinomial model. Both of them rely on the *Naïve Bayes assumption*, which means that the terms are independent, in terms of probability, given the class.

The Naïve Bayes belongs to the probabilistic classifiers framework. In it, the CSV computed for the document d_j and the category c_i is the probability $p(c_i|d_j)$. For the case of the Naïve Bayes, the probabilities $p(d_j|c_i)$ are computed, and using Bayes' Theorem (see chapter 2, section 2.1.1.2), the final $p(c_i|d_j)$ are given by $p(c_i|d_j) = p(c_i)p(d_j|c_i)/p(d_j)$. All the probability notation used here is defined on chapter 2.

In this model a document is an ordered sequence of terms drawn from the same vocabulary, and the Naïve Bayes assumption here means that the occurrences of the terms in a document are conditionally independent given the class, and the *positions* of these terms in the document are also independent given the class[1]. Thus, each document d_j is drawn from a multinomial distribution of words with as many independent trials as the length of d_j. Then,

$$p(d_j|c_i) = p(|d_j|)\frac{|d_j|!}{\prod_{t_k \in d_j} n_{jk}!} \prod_{t_k \in d_j} p(t_k|c_i)^{n_{jk}} , \tag{1.1}$$

where t_k are the distinct words in d_j, n_{jk} is the number of times the word t_k appears in the document d_j and $|d_j| = \sum_{t_k \in d_j} n_{jk}$ is the number of words in d_j. As $p(|d_j|)\frac{|d_j|!}{\prod_{t_k \in d_j} n_{jk}!}$ does not depend on the class, we can omit it from the computations, so that we only need to calculate

$$p(d_j|c_i) \propto \prod_{t_k \in d_j} p(t_k|c_i)^{n_{jk}} . \tag{1.2}$$

The estimation of the term probabilities given the class, $\widehat{p}(t_k|c_i)$, is usually carried

[1] The length of the documents is also assumed to be independent on the class.

out by means of the Laplace estimation:

$$\widehat{p}(t_k|c_i) = \frac{N_{ik}+1}{N_{i\bullet}+M}\,, \tag{1.3}$$

where N_{ik} is the number of times the term t_k appears in documents of class c_i, $N_{i\bullet}$ is the total number of words in documents of class c_i, i.e. $N_{i\bullet} = \sum_{t_k} N_{ik}$, and M is the size of the vocabulary (the number of distinct words in the documents of the training set).

The estimation of the prior probabilities of the classes, $\widehat{p}(c_i)$, is usually done by maximum likelihood, i.e.:

$$\widehat{p}(c_i) = \frac{N_{i,doc}}{N_{doc}}\,, \tag{1.4}$$

where N_{doc} is the number of documents in the training set and $N_{i,doc}$ is the number of documents in the training set which are assigned to class c_i.

The multinomial Naïve Bayes model can also be used in another way: instead of considering only one class variable C having n values, we can decompose the problem using n binary class variables C_i taking its values in the sets $\{c_i, \overline{c}_i\}$. This is how we get from multilabel to binary problems as explained in section 1.2.4.

In this case n Naïve Bayes classifiers are built, each one giving a posterior probability $p_i(c_i|d_j)$ for each document. In the case that each document may be assigned to only one class (single-label problems), the class $c^*(d_j)$ such that $c^*(d_j) = \arg\max_{c_i}\{p_i(c_i|d_j)\}$ is selected. Notice that in this case, as the term $p_i(d_j)$ in the expression $p_i(c_i|d_j) = p_i(d_j|c_i)p_i(c_i)/p_i(d_j)$ is not necessarily the same for all the class values, we need to compute it explicitly through

$$p_i(d_j) = p_i(d_j|c_i)p_i(c_i) + p_i(d_j|\overline{c}_i)(1-p_i(c_i))\,.$$

This means that we have also to compute $p_i(d_j|\overline{c}_i)$. This value is estimated using the corresponding counterparts of eqs. (1.2) and (1.3), where

$$\widehat{p}(t_k|\overline{c}_i) = \frac{N_{\bullet k} - N_{ik} + 1}{N - N_{i\bullet} + M}\,. \tag{1.5}$$

N is the total number of words in the training documents and $N_{\bullet k}$ is the numbers of times that the term t_k appears in the training documents, i.e.

$$N_{\bullet k} = \sum_{c_i} N_{ik} \, .$$

There is a computational issue related to this model of a certain importance, that should be clarified. In eq. (1.2) the term $\prod_{t_k \in d_j} p(t_k|c_i)^{n_{jk}}$ needs to be computed logarithmically, in order to avoid numeric problems. In fact, if we are dealing with a multiclass problem, we do not need to return to non-logarithmic space, and we can just return the category with greater $\log p(d_j|c_i)$ value.

1.3.1.2 The Rocchio Method

The Rocchio method [110] is a categorization model, coming from the Information Retrieval field, adapted from the framework of relevance feedback. It is a very simple model and therefore, it is very used for comparison purposes.

It relies heavily on a free interpretation of the cluster hypothesis (proposed by van Rijsbergen [108]):

Hypothesis 1 (Cluster hypothesis)**.** *Closely associated documents tend to be relevant to the same requests.*

The Rocchio model (see, for instance [59]) adapted to Text Categorization assumes that *closely associated documents tends to belong to the same category.* This is applied to the Rocchio model building the centroid of the documents of each category and testing, for each unlabeled document, which group is near. In order to be more realistic, the final centroid is built from two previously built centroids: the *positive* centroid (the average of the documents of the category), and the *negative* (average of the documents not belonging to that category). We choose a point in the space with the balance of being close to the positive centroid but far from the negative one.

Numerically, we can express the learning process as the computation of n

centroids h_i:

$$h_i = \frac{\beta}{|\{d_j \in c_i\}|} \sum_{d_j \in c_i} \frac{d_j}{\|d_j\|} - \frac{\gamma}{|\{d_k \notin c_i\}|} \sum_{d_k \notin c_i} \frac{d_k}{\|d_k\|} \tag{1.6}$$

Where $\|d_j\|$ means the Euclidean norm of vector d_j, and $|\{d_j \in c_i\}|$ is the number of documents which belong to category c_i. β and γ are positive real parameters whose values are dependent on the collection (even $\beta = \gamma$ can be a good election). For these computations, d_j document vectors are built normally following the tf-idf scheme, explained before.

In the final step of the computation of centroids, the components of h_i which are negative are set to zero. After this, a soft categorization classifier g can be defined as the dot product between document and centroid vector, both normalized:

$$g(d_j, c_i) = \frac{d_j \cdot h_i}{\|d_j\| \|h_i\|} = \frac{\sum_k d_{jk}\, h_{ik}}{\sqrt{\sum_k d_{jk}^2}\ \sqrt{\sum_k h_{ik}^2}}$$

Where d_{jk} and h_{ik} represents the k-th coordinate of the vectors d_j and h_i, respectively.

Clearly this is equivalent as measuring the cosine of the angle between d and h_i, which is very used as a dissimilarity measure (a substitute for "distance") in Information Retrieval. Moreover, knowing that d and h_i are vectors with all the coordinates positive, it holds that $g(d, c_i) \geq 0, \forall d \in \mathcal{D}, \forall c_i \in \mathcal{C}$. So, the CSV of any vector and category lies on the interval $[0, 1]$.

Although eq. (1.6) is the original formula to build the centroid vector, we should note some remarks. First of all, h_i vector is always normalized to compute the value $g(d, c_i)$, and then, only one parameter should be needed:

$$\begin{aligned} h_i &= \frac{\beta}{|\{d_j \in c_i\}|} \sum_{d_j \in c_i} \frac{d_j}{\|d_j\|} - \frac{\gamma}{|\{d_k \notin c_i\}|} \sum_{d_k \notin c_i} \frac{d_k}{\|d_k\|} \\ &= \beta \left(\frac{1}{|\{d_j \in c_i\}|} \sum_{d_j \in c_i} \frac{d_j}{\|d_j\|} - \frac{\gamma}{\beta} \frac{1}{|\{d_k \notin c_i\}|} \sum_{d_k \notin c_i} \frac{d_k}{\|d_k\|} \right) \end{aligned}$$

Thus, the only required parameter is the proportion γ/β, because the role of β is just to be a scaling factor for the vector (and this transformation is lost when doing normalization). There are some works with nice results which find optimum values for these parameters, trying to optimize an evaluation measure [68; 95].

The second remark is that the value of γ (and hence, the γ/β one) can be set to 0. This results in a classifier which only measures the distance to the positive centroid of the category (and no extra parameters are needed). This classifier has been named sometimes in the literature as the *Find Similar* method (see, for instance [46]). This Find Similar classifier is very close to the Centroid classifier method, but it is not exactly the same. In Centroid classifier, the centroid vector is often obtained by adding all the unnormalized vectors of the category, and then normalizing the centroid by its Euclidean norm. In Rocchio and Find Similar methods, the centroid is built adding normalized vectors, and dividing by the number of added vectors. In both cases, the resulting centroid is a normalized vector and the classification procedure is the same (a dot product between unit vectors), but the way the centroid vectors are built is different.

1.3.1.3 Linear Support Vector Machines

Support Vector Machines are relatively new classifiers. Introduced in 1992 by Boser, Guyon and Vapnik [8], they are based on some Statistical Learning Theory principles (see [128] for more details on this area). These methods have been very popular in many fields (bioinformatics, text, handwriting recognition, etc).

Linear Support Vector Machines are linear classifiers, which is a set of more general methods. A linear classifier makes a decision based on the lineal combination of feature values, that is to say, takes this form:

$$f(d) = g(d \cdot w)$$

Where w is a vector of "weights", and d is the feature vector, both $w, d \in \mathbb{R}^n$. In this notation, $v_1 \cdot v_2$ means the dot product of vectors v_1 and v_2.

For the specific case of linear SVMs, f is:

$$f(d) = \text{sign}(d \cdot w - b)$$

Where sign(x) function equals to one iff $x > 0$, equal to minus one iff $x < 0$ and is zero for $x = 0$. $b \in \mathbb{R}$ is a weight called the bias.

Ignoring the case $d \cdot w = b$, it is obvious that f is a map to $\{-1, 1\}$. In Support Vector Machine notation, a binary problem is identified with the set of labels $\mathcal{C} = \{-1, 1\}$ (where -1 plays the role of $\overline{c}$, and 1 is c).

The geometric interpretation of this classification rule is very simple. w can be seen as the normal vector of a (n-dimensional) hyperplane. This hyperplane divides the space into two subsets. Those points which lie on the part of the space where the normal is pointing to, will be classified with a positive label (1), and on the contrary they will be mapped to -1. A 2-dimensional example of how a hyperplane separates data can be seen graphically in figure 1.2:

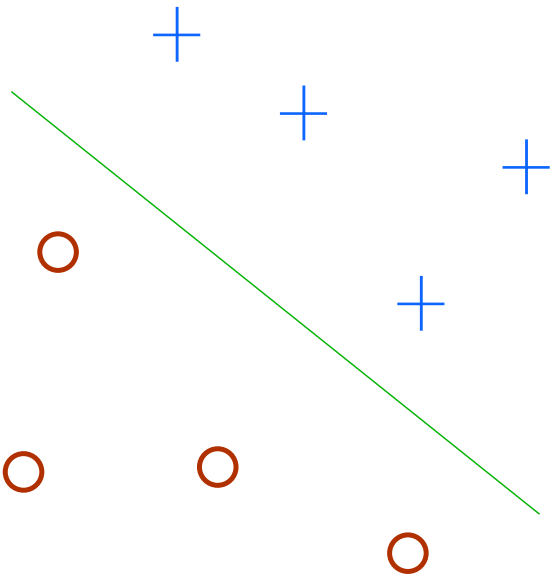

Figure 1.2: How a hyperplane can separate two different sets of data.

Until this point we know how to categorize data with a learnt SVM, but we left the three following issues unanswered:

1. How is this hyperplane learnt?

2. If the data is separable, which hyperplane should we choose?

3. What if the data is not separable?

We shall try to solve the first question giving a method to learn SVM, based on the answer to the second one (i.e. which hyperplane is better). The case of non-separable data will be reviewed afterwards, because it is a extension of the separable one.

So, let us assume that our data is separable. Assuming certain training data, a "good" hyperplane should be capable to assign the training data its own label when being used to classify it. From a geometrical point of view, a good hyperplane should perfectly separate the elements from the two categories, $\{-1, 1\}$. In certain occasions, there are several hyperplanes which are perfect solution to this problem, as seen, for example in figure 1.3 (where w_1, w_2 and w_3 are good candidates).

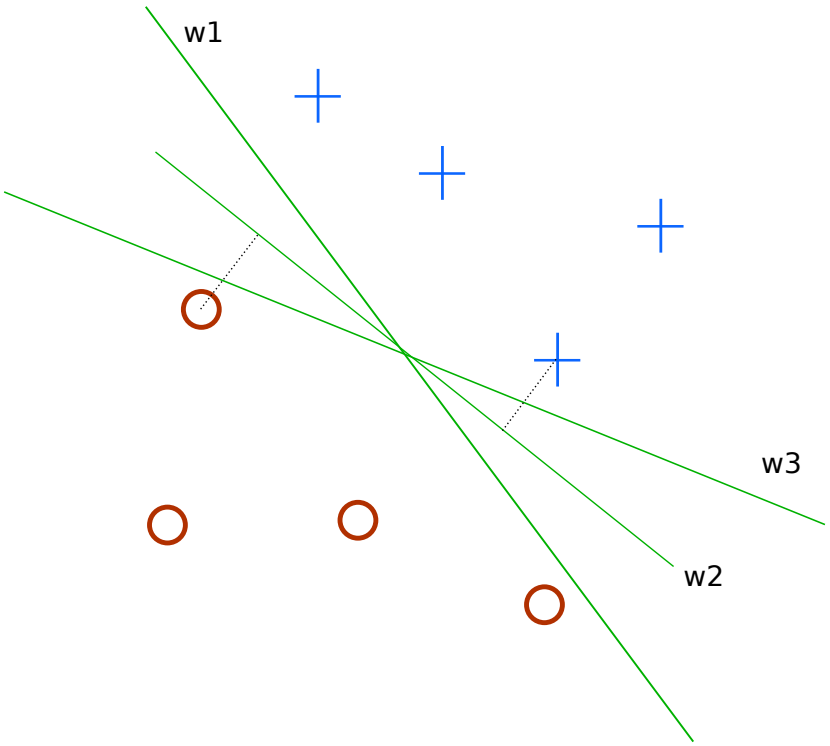

Figure 1.3: Several hyperplanes that perfectly separate the same set of examples.

A classic algorithm to obtain a separating hyperplane (assumed that the data is separable) is Rosenblatt's primal perceptron algorithm [111], which is guaranteed to converge. By using this method, we obtain one particular (random) hyperplane, among the valid solutions to this problem, with no particular setting.

The SVM learning algorithm pursues the maximization of the margin, that is to say, the decision boundary should be as far away from the data as possible. In figure 1.3, both w_1 and w_3 hyperplanes are too close to one data point. However, w_2 seems to be at a reasonable distance, and it is an optimum solution using this criterion. This is what we intuitively call the margin. Let us define it analytically.

Let be w the optimum hyperplane. Assume l training points. All of them should then verify $y_i(wx_i + b) > 0$. We define $\gamma_i = \frac{1}{\|w\|}(wx_i + b)$, as the distance of the i-th point, x_i to the hyperplane solution. Given a certain hyperplane $wx+b = 0$, we define the *margin* γ as $\gamma = \min_i \gamma_i$. We formulate the optimization problem:

$$\textit{Find } w, b, \textit{ which maximize } \gamma \textit{ such that } y_i \tfrac{1}{\|w\|}(wx_i + b) \geq \gamma, \forall i = 1, \ldots, l$$

Note that w, b of hyperplane $wx+b = 0$, solution to the optimization problem, can be scaled by a real positive constant. We then choose $\|w\| = \frac{1}{\gamma}$. Given the previous chosen value of $\|w\|$, and knowing that $\min \|x\| \Leftrightarrow \min \frac{1}{2}\|x\|^2$, for all vector x, we reformulate the problem as a minimization one:

$$\textit{Find } w, b, \textit{ which minimize } \tfrac{1}{2}\|w\|^2 \textit{ such that } y_i(wx_i + b) \geq 1, \forall i = 1, \ldots, l.$$

The problem can be written on an unconstrained form, using the set $\alpha = \{\alpha_i, i = 1, \ldots, l, \alpha_i > 0\}$ which is called the set of Lagrange multipliers. The resulting problem is then:

$$\min_{w,b} \left\{ \frac{1}{2} \|w\|^2 + \text{penalty} \right\} = \min_{w,b} \left\{ \frac{1}{2} \|w\|^2 + \sum_{i=1}^{l} \max_{\alpha_i \geq 0} \alpha_i (1 - y_i(wx_i + b)) \right\}.$$

Taking out the maximum, and swapping max and min

$$\max_{\alpha_i \geq 0} \min_{w,b} \left\{ \frac{1}{2} \|w\|^2 + \sum_{i=1}^{l} \alpha_i (1 - y_i(wx_i + b)) \right\}.$$

We define, for a certain hyperplane, and given a fixed set of multipliers, $J(w, b; \alpha) = \frac{1}{2}\|w\|^2 + \sum_{i=1}^{l} \alpha_i (1 - y_i(wx_i + b))$, then, we obtain the partial derivatives equal to zero in order to perform optimization:

$$\frac{\partial J(w, b; \alpha)}{\partial w} = w - \sum_{i=1}^{l} \alpha_i y_i x_i = 0$$

$$\frac{\partial J(w, b; \alpha)}{\partial b} = -\sum_{i=1}^{l} \alpha_i y_i = 0$$

We perform substitution inside the "min" expression obtaining what is called the *dual* problem:

$$\max_{\substack{\alpha_i \geq 0 \\ \sum_{i=1}^{l} \alpha_i y_i = 0}} \left\{ \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) \right\}. \tag{1.7}$$

This problem can be solved by traditional optimization procedures, and when the optimal elements α are found, then w and b are computed, finding the desired hyperplane solution.

The non-separable case, solved by Cortes and Vapnik [31], only introduces some positive *slack variables* ξ_i, such that:

$$\begin{array}{ll} wx_i + b \geq 1 - \xi_i & \text{if } y_i = 1 \\ wx_i + b \leq -1 + \xi_i & \text{if } y_i = -1 \\ \xi_i \geq 0, \forall i = 1, \ldots, l & \end{array}$$

This is known as the soft-margin extension. In this extension, instead of just minimizing $\frac{1}{2}\|w\|^2$, we find the minimum of $\frac{1}{2}\|w\|^2 + C(\sum_{i=1}^{l} \xi_i)^k$, where C is a constant, chosen by the user, which represent the cost or penalty of errors, and k is a positive number. With this problem setting, and $k = 1$ or 2, the resulting optimization problem is also a quadratic problem. $k = 1$ also has the advantage that neither the slack variables, ξ_i, nor their Lagrange multipliers appear on

the dual problem (see [12] for more details), resulting on a similar optimization problem, similar to that presented on eq. (1.7):

$$\max_{\substack{0\leq\alpha_i\leq C \\ \sum_{i=1}^{l}\alpha_i y_i=0}} \left\{ \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) \right\}. \tag{1.8}$$

Problem in eq. (1.8) is almost the same problem as the one shown in eq. (1.7), but giving an upper bound to each α_i. Thus, it can be solved using the same optimization algorithms, and the solution is also given using optimum α_i to find w and b.

For a more detailed discussion on how this classifier is motivated, and what optimization algorithm can be used to find the optimum α_i values, we refer the reader to [12; 128].

1.3.2 Other Relevant Approaches to Build Classifiers

There are other notable approaches to build text classifiers, which have been used on the literature. We list here some of the most cited references, although we shall not be using any of these methods on the following chapters.

- *Manually built rule systems*: these methods rely on a set of logic rules, in disjunctive normal form, with the following pattern:

$$\text{IF } \left(\bigvee_i \left(\bigwedge_j t_{ij} \in d \right) \right) \text{ THEN classify } d \text{ as } c$$

  where $t_{ij} \in \mathcal{T}$. The set of rules was manually specified by human experts[1]. Obviously, a different set of rules needs to be specified for each corpus. This is the case of the CONSTRUE system [52] (built for the Reuters collection),

- *Automatically inducted rule systems*: these classifiers take the same form than the previous, but some automatic data mining methods are used to

[1] Observe that we used the term *automated* in our definition of Text Classification (in section 1.1), with the meaning that a classifier was an algorithm capable of labeling documents. However, nothing keep us from manually building that algorithm, as in this case.

build the set of rules. Some examples of this approach are the WASP-1 [5], CHARADE [97] and RIPPER [24] systems.

- Models based on *decision trees*. Those methods use some well-known decision tree learning algorithms (ID3, C4.5, C5, etc), to infer a classifier from training data. In order to make this task easier, a term selection procedure is often applied, to reduce the dimensionality of the dataset.

  Some experiments with classical decision trees algorithms as comparison methods are found on [60; 77; 134]. There are few papers on decision trees to categorize text where the procedure is innovative (for example in [130], a decision tree is combined with a boosting algorithm) being in almost all publications a comparative procedure used to be tested against a new classifier.

- Other *probabilistic* models: apart from the multinomial Naïve Bayes, presented in section 1.3.1.1, there are other probabilistic classifiers.

  Among the alternative Naïve Bayes approaches, the work by Lewis [76] is a good compilation of several uses of this model in Information Retrieval and Text Categorization. Other important approaches that use the Naïve Bayes are [106] and [64].

  Beyond this well known model, we have the limited dependence classifier by Sahami [113] which has been presented in some occasions [46] as a "Bayesian Network classifier". This is a very powerful model which often improves Naïve Bayes results.

  Among the general Bayesian network classifier, the work by Klopotek [66] is clearly innovative, and is an isolated reference on this sub-area.

- General *Support Vector Machines* based models: on this area are remarkable the first works by Joachims [60], where several polynomial and radial basis function based models were tested on Reuters and Ohsumed. A deeper reference on the usage of SVMs in the problem of Text Categorization is perhaps its book [61].

In all the relevant publications on this sub-area it is very difficult to find a work where the difference in using a linear SVM (explained in section 1.3.1.3) and a polynomial (with degree greater than 1) or another kernel, makes a huge improvement on the results, which is computationally worthwhile.

- *Memory based classifiers*, being the more representative the k-NN. It is a classic approach on Machine Learning, and has been used as a comparison in lot of works (see, for instance [60; 130; 135]).

  To label an unknown document d, the k-NN algorithm matches the vector of the document to classify with all the documents in a training set. Then, it computes the k-nearest neighbors (this is why the method is called k-NN) to predict the label of the document to be classified. In this algorithm, k is a free parameter, and its optimum value can be set by doing previous experimentation on the training set (see, for instance, [78]).

  In order to compute the distance (dissimilarity), the usual approach used is the cosine-similarity [108] between documents. The k-NN method has linear "learning" time, but its classification time increases linearly with the number of training samples.

- Other methods, based on *neural networks* (see, for instance [112; 132]), or *regression methods* (like the LLSF classifier [133]).

1.4 Difficulties of the Problem

Text Document Categorization implies several difficulties [61] that can be summed up in the following five:

1. The *high dimensionality of the characteristic space* (variables) presented in documents (one variable for each term). If each term is identified with one variable, it is not very difficult to find problem instances with thousand of characteristics, which contrasts with "classic" Machine Learning problems (where the number of variables is low, often under 100). This high

dimensionality can lead to two kinds of problems: estimation (when more parameters have to be estimated, less confidence on the model) and computational (several classic models used on Machine Learning are not easily applicable here due to its poor performance on so large problem instances).

2. *Sparse vectors.* Although the representation of a document as a vector is very similar to the classic pattern representation in traditional classifiers, this vector will be very sparse. This is because a document only contains a small subset of words from the whole lexicon (the set of all words of the collection of documents).

3. *Heterogeneity.* Due to phenomena like polysemy (several different meanings for a term) and synonymy (several different terms with the same meaning), two documents of the same category could have no common terms, and two documents of different categories could be composed of a very similar set of terms.

4. *Ambiguity on the labeling process.* This is a difficulty inherent to this kind of systems. If we build a news classifier and we receive some (unlabeled) news article about David Beckham, it could be labeled as "sports" or "celebrity news" (even in the circumstances that it is not allowed, we should be able to categorize documents in more than one class). On the other hand, one article about the results of the last Euroleague Basketball could be first classified into "sports", or more specifically, into "basketball" (this is a case where the categories show a certain hierarchy).

5. The *evaluation* procedures for classifiers, assumed a multilabel or hierarchical problem, should not be only those used on traditional classification (F_1, breakeven point, ...), but also include other standard measures which take into account hierarchy or relationships among categories.

1.5 Evaluation of Classifiers

In this section we review the most common approaches to the evaluation of Text Classification. Most of these measures have been used in experimentation sections

	assigned c	assigned $\overline{c}$
$d \in c$	TP	FN
$d \in \overline{c}$	FP	TN

Table 1.1: Contingency table for binary classification

on this work, and therefore we explain them carefully. We distinguish between two different evaluation methodologies: *category* and *document centric*.

Category centric seem to be the most common measures. In them, a category is assigned a list of documents, and this is what we evaluate (each document is a correct or wrong assignment). Document centric measures are less common, but also used. They look at the category assignments for each document, and measures are computed based on it.

1.5.1 Category-centric Measures

Let us assume first that we have a binary classification problem. On classifying a single document, we find four different scenarios (we assign the document to the category or not, and the document really belongs or not to it). This results in two good assignments and two different kinds of errors.

Let us review these scenarios: from the category point of view, we can make assignments of documents which really belong to the category (producing *true positives* or TP), or assigning documents which are not in the category (giving *false positives* or FP). Conversely, we can say that a document does not belong to the category, finding afterwards that this fact is real (resulting in a *true negative* or TN), and finally, not assigning the category to documents which really belong to it (being this a *false negative* or FN).

For clarity, we show the contingency table 1.1, where we have counted all those quantities in this binary classification problem ($\mathcal{C} = \{c, \overline{c}\}$).

One initial method to compute how good is a classifier would be using the *accuracy*. It is a measure that computes the proportion of "good assignments". For the case of a binary classifier (showed in previous contingency table 1.1), accuracy (ACC) can be defined as follows:

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}$$

However, accuracy is not very good to measure the performance of a text classifier. The reason for this, pointed by Yang [135], is that the large value of the denominator on these problems makes the measure more insensitive to the variation of $(TP+TN)$ than other approaches. Instead, a good way to summarize these four quantities is to compute the following evaluation measures:

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

The *precision* can be seen as a measure of *purity* of the results, that is to say, how many good results are there among all the assignments. The *recall*, on the other hand, represents the degree of *completeness* of the true labelings. Note that we can easily get recall = 1 if we assign all the documents to the category (precision would be low due to the false positives). Conversely, with a few (and true) assignments we could get a high precision (even 1), but with a low recall (due to the abundance of false negatives). From now on, we shall use "prec" for precision and "rec" for recall.

An ideal classifier should obtain both a precision and a recall equal to 1. We shall use a measure that combines these measures in one, giving us an idea of how good is the classification. This measure is the Rijsbergen's F_α-measure, concretely the F_1 ($\alpha = 1$) which is the harmonic mean between precision and recall. We present several equivalent forms of this measure:

$$F_1 = \frac{2 \text{ prec rec}}{\text{prec} + \text{rec}} = \frac{2\,TP}{2\,TP + FN + FP}$$

The F_1 measure is the perfect candidate if we are doing hard categorization, because we can compute the precision and recall measures easily with this kind of decisions (the document belongs or not to the category). But, what measure should we choose if we are in a problem of soft categorization? For the binary base, we have, for each document d_j a CSV value $g(d_j, c)$. If we sort that list in

decreasing CSV and we traverse it from the beginning, measuring precision and recall at each point, it is a fact that recall will always grow or maintain its value. On the other hand, after the first true positive found, the precision will decrease or keep its value[1]. We define the *precision-recall breakeven point* as the point in which precision equals to recall. If no such point exists, the average between the two closest values of precision and recall is computed instead.

If we have several categories (i.e. $\mathcal{C} = \{c_1, \ldots, c_p\}, p > 2$), there are two ways to combine measures among categories: *micro* and *macroaveraging*. Essentially, *microavegaging* consists of computing the measures considering a global contingency table (adding all individual contingency tables for every category), and *macroaveraging* is the average of each measure, first computed by class. A microaveraged measure is determined by the values obtained in very common categories (because of their higher contribution to the global contingency table), and a macroaveraged one takes more into consideration rare categories (because all of them are treated equal).

Thus, we could have the micro/macro averaged versions of the precision/recall breakeven point and the F_1, for soft and hard categorization (respectively) of multiclass (or multilabel) classifiers. Let TP_i, TN_i, FP_i be the true positives, true negatives and the false positives of category c_i respectively. Then, the microaveraged F_1 (μF_1) and the macroaveraged F_1 (MF_1) can be computed as follows:

$$\mu F_1 = \frac{2\sum_i TP_i}{\sum_i (2\,TP_i + FN_i + FP_i)}, \quad MF_1 = \frac{1}{|\mathcal{C}|}\sum_i \frac{2\,TP_i}{2\,TP_i + FN_i + FP_i}$$

The precision recall breakeven point measure can also be micro or macroaveraged. In both cases, we traverse the sorted list of all $g(d_j, c_i)$ values, joined for all c_i categories. For the microaveraged case, every new true positive, negative or false positive found is added to recompute a "global" precision and recall (and we then, stop this process when precision equals to recall). In the macroaveraged case, the average of individual precision/recall breakeven point values for each category is computed.

[1]Precision and recall decrease and grow monotonically, respectively.

1.5.2 Document-centric Measures

With these measures, we can also measure, from the document point of view, how categories are assigned to each document. This is a method inspired in Information Retrieval systems and in their measures [108], where a document is considered a query, the assigned categories are the retrieved documents, and the true categories are the relevance judgments.

For this case, we can also define precision and recall. *Precision* would be the number of true retrieved categories divided by the number of retrieved categories, and *recall* would be the number of true retrieved categories divided by the total number of true categories. We can also compute a F_1 measure with those new definitions of precision and recall.

If we are dealing with a huge number of categories, and we are trying to help a human indexer, instead of using the whole list of retrieved categories, we can compute precision and recall at a certain number of retrieved categories. So, we could have, for instance, prec@5, rec@10, F1@5,... And for all, we can have micro and macroaveraged versions (following previous definitions).

A way to summarize precision and recall, computed from the point of view of the documents, is the *medium average precision on the 11 standard points of recall* (MAP). It has been used by Yang [135] for Text Categorization problems, and is a very well known measure in the Information Retrieval community (almost every evaluation package includes it). The procedure to compute this measure is the following: a threshold is repeatedly tuned so as to allow recall to take up values of $\{0.0, 0.1, \ldots, 0.9, 1\}$, and precision is computed for these 11 values of threshold, and afterwards averaged over those resulting values.

1.6 A Review of Several Testing Corpora

In this section we make a small review of typical corpora used in the supervised Text Categorization problem. These corpora have been prepared for flat Text Categorization. In more advanced problems of categorization we shall be using several specific corpus which will be described in that chapter.

1.6.1 Reuters-21578

This is a collection of documents, compiled by David D. Lewis (see, for instance [77]), that appeared on the Reuters agency newswire in 1987. The documents were assembled and indexed with categories. The documents were marked with SGML, which is often removed for experimentation.

The corpus is named after the number of documents it contains (a total of 21578), and has several standard splits. The most used one, the ModApte, divides the set of documents into a training and a test set. Categories only assigned to documents in the test set are removed, and the resulting number of categories obtained is 90.

This is the de facto collection to make text categorization. It has a decent number of categories, it is a multilabeled corpus, it has been exhaustively tested, and it is an unbalanced problem (there are categories with many documents and other with few).

There are several authors which take a subset of the Reuters collection, of documents labeled with the 10 most popular categories. However, it has been proved [34] that this election is unfair, and results on a very easy corpus, where even bad algorithms could obtain very good results.

1.6.2 Ohsumed

The Ohsumed collection [55] is a set of 348566 references from MEDLINE, the online medical information database collected by William Hersh. For every record, coming from one of 270 medical journals over a five-year period (1987-1991), we are given some data and the assigned MeSH term, which is the assigned category (see chapter 4).

Because the number of categories of the MeSH thesaurus is huge, it is often chosen a subset of 23 categories (*heart diseases*), which are the root of some categories in a hierarchy. Documents which do not belong to that subtree of categories, are discarded. This is the methodology followed in [60]. It is a multilabel corpus.

The available fields are title, abstract, MeSH indexing terms, author, source, and publication type. The traditional split used for experiments consist in taking years 1987-88 for training, and 1990-91 for testing purposes.

It should be noted that the National Library of Medicine has agreed to make the MEDLINE references in the test database available for scientific experimentation.

1.6.3 20 Newsgroups

20 Newsgroups, collected by Ken Lang [71], is a collection of about 20000 UseNet news postings into 20 different newsgroups (which act as categories). It is almost a multiclass corpus, but about 10 percent of the articles are crossposted (they belong to two different categories). One of the common evaluation procedures for this corpus is to give the percentage of correctly assigned documents, having into consideration the fact that a crossposted article counts positive in any of its categories (but not in both).

There is no standard split to work with it, and then, a random split or a cross validation scheme is needed to perform experiments.

1.6.4 RCV1 corpus

RCV1 corpus is a relatively more recent corpus also proposed by Lewis [78], based on Reuters news stories. It contains about 35 times as many documents (806791 for the final version, RCV1-v2) as the Reuters-21578. The documents are preprocessed, with stopwords removed, and terms already stemmed. The terms are also unsorted (in order to avoid document reconstruction and legal issues).

The number of categories ("topic codes") is 103 (after the split). The corpus includes other two disjoint set of categories called "industry codes" (870) and "region codes" (366). The second two are less used than the first one as labels in categorization experiments.

A standard split is also provided, to avoid the lack of a partition which happened in the old Reuters corpus. The called LYRL2004 split gives a training set

with over 23000 documents, and a test set with 781000. Note that the dimensions are more updated to the actual problems, rather than the numbers of the old corpus.

Chapter 2

Probability Theory and Bayesian Networks

2.1 Probability Theory

2.1.1 Basic concepts

We review here some basic concepts of probability theory for the *discrete* case. Most of the definitions here have been taken or adapted from [99]. For a more extensive treatment on this subject, including the continuous case we refer the reader to [107].

2.1.1.1 Probability Function and Probability Spaces

A *sample space* is a countable[1] set $\Omega = \{x_1, x_2, \dots\}$, which represents the set of all possible outcomes of an experiment.

A function $P : 2^{\Omega} \longrightarrow \mathbb{R}$, (where 2^{Ω} is the set consisting of all the subsets, called the events, of Ω) is called a *probability function* if satisfies the following assumptions (Kolmogorov's probability axioms):

1. $0 \leq P(A) \leq 1, \forall A \subseteq \Omega$.

2. $P(\Omega) = 1$.

[1] We shall refer to a set as *countable* if it has the same cardinality (number of elements) as some subset of the set of natural numbers, $\mathbb{N}$.

3. If $E_1, E_2, \ldots$ is a countable sequence of mutually disjoint events (i.e. $E_i \cap E_j = \emptyset, \quad \forall i \neq j$), then it holds:

$$P\left(\cup_i E_i\right) = \sum_i P(E_i).$$

If P is a probability function, then the pair (Ω, P) is called a *probability space*.

2.1.1.2 Conditional Probability and Bayes Theorem

Let be $A, B \subseteq \Omega$, such that $P(B) \neq 0$. Then, the *conditional probability* of A given B, noted $P(A|B)$ is given by:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

We say that two events A, B, where $P(A) \neq 0$, and $P(B) \neq 0$ are *independent* if $P(A|B) = P(A)$.

Two events A, B, where $P(A) \neq 0$, and $P(B) \neq 0$ are *conditionally independent* given C if $P(A|B \cap C) = P(A|C)$.

If $E_1, \ldots, E_n$ is a set of mutually disjoint events such that $\cup_i E_i = \Omega$ (i.e. they are a *partition* of the sample space), and $P(E_i) > 0, \forall i = 1, \ldots, n$, then, for any event A:

$$P(A) = \sum_{i=1}^{n} P(A \cup E_i) = \sum_{i=1}^{n} P(A|E_i)P(E_i). \tag{2.1}$$

This property is called the *total probability law*, and can be proven with basic set theory.

Theorem 2.1.1. (Bayes) *Given two events $A, B \in \Omega$, such that $P(A) \neq 0$ and $P(B) \neq 0$, we have*

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}.$$

2.1.1.3 Random Variables

A (discrete) *random variable* is a function that maps events to values of a countable set (for example, the integers), with each value in the range having probability

greater than or equal to zero. We shall represent a random variable with a capital letter, and the value where events are mapped, as lowercase letters.

A random variable induces a (new) probability function over its values. We shall use the notation $X = x$ to represent the set of events of Ω mapped, by X, to x. If we define $P_X(\{x\}) := P(X = x)$, then P_X is a probability function. For the sake of convenience, we shall not use the P_X notation, and we shall simply write $P(X = x)$, which will be called the *probability distribution* of X (or just the *distribution* of X). If we know that X is a random variable we could directly name the distribution of X as $P(X)$ with no ambiguity.

Sometimes, given a certain random variable X and x one of its values, we shall write, for brevity, $p(x)$ instead of $P(X = x)$. This quantity will be referred as the *probability of* x.

For a certain random variable A, we shall call the set of values A can have, the "range of A" (and sometimes will be noted as Ω_A).

Given two random variables, X, Y, defined on the same sample space, and two values x of X and y of Y, we can compute the probability of the intersection event (the elements both mapped to x via X and to y via Y). We shall note the probability of this intersection event as $P(X = x, Y = y)$. This will be called the *joint probability distribution* of X and Y. We can also define a joint probability distribution of an arbitrary set of random variables $\{X_1, X_2, \ldots, X_n\}$ as $P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$.

Now, we shall introduce the operation that is called the *marginalization*. Given a joint probability distribution $P(X = x, Y = y)$, we can obtain the distribution $P(X = x)$ (called the *marginal probability distribution*[1] of X) by summing on all y values in the range of Y, i.e.:

$$P(X = x) = \sum_y P(X = x, Y = y).$$

This is another expression for the law of total probability, presented on eq. (2.1).

[1]The term "marginal" is used when a distribution is obtained by marginalization of another joint distribution, but in fact it is also a "conventional" distribution.

Finally, understanding the analogy between random variable and events, we can also define the *conditional probability distribution* of a random variable X given Y as:

$$P(X = x|Y = y) = \frac{P(X = x, Y = y)}{P(Y = y)}.$$

Again, the case can be extended to a set of two or more variables. A very interesting result that permits the calculation of the joint distribution of a set of random variables using only conditional probabilities is called the *chain rule*:

$$P(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i|X_{i-1} = x_{i-1}, \ldots, X_1 = x_1),$$

which can be easily proven by direct application of Bayes' Theorem.

2.1.1.4 Conditional Independence and Observations for Random Variables

Two of the "classic" notions on probability theory, already mentioned on section 2.1.1.2 are the concept of *independence* and *conditional independence*.

From the point of view of random variables, we can rewrite the definition of *independent events*. We say that the two events $X = x$ and $Y = y$ (being X, Y two random variables, and x, y two values of X, Y) are independent if:

$$p(x, y) = p(x)\, p(y).$$

We shall also need the following definition:

Two random variables X and Y are *independent* if and only if

$$P(X = x, Y = y) = P(X = x)\, P(Y = y),$$

for all x, y values of X and Y, respectively.

For problems with at least three random variables we can observe sometimes the phenomenon of *conditional independence*. Let us have three random variables X, Y and Z. We shall say that X and Y are *conditionally independent given* Z if

and only if every pair of values x (of X) and y (of Y) is conditionally independent for each z (of Z) such that $p(z) > 0$, that is to say:

$$\forall x, y, z, \quad p(z) > 0 \;\Rightarrow\; p(x, y|z) = p(x|z)\, p(y|z).$$

This concept is not limited to just three variables, and it is extensible to larger sets of variables.

We shall say that a variable is *observed* if is set to one of its values of its range.

If we want to test *independence with respect to an observed variable*, we only need to test independence with respect to that assigned value. Note that observing a variable can change the independence relationships among a set of variables.

We shall note that a variable A is independent of other variable B, after the observation of C with $A \perp B|C$. If A is independent of B (without any additional observation), we shall just write $A \perp B$.

Two very interesting properties of the independence relationships are the following. Given $\mathbf{W}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$, sets of random variables,

- (symmetry) $(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z}) \Rightarrow (\mathbf{Y} \perp \mathbf{X}|\mathbf{Z})$.
- (decomposition) $(\mathbf{X} \perp \mathbf{Y}, \mathbf{W}|\mathbf{Z}) \Rightarrow (\mathbf{X} \perp \mathbf{Y}|\mathbf{Z})$.

We shall not give the proof here. Instead, we refer the reader to [102].

2.2 Bayesian Networks

2.2.1 Motivation

Let be a set of random variables $X_1, \ldots, X_n$. The joint distribution $P(X_1, \ldots, X_n)$, in the general case, needs to store an exponential number of parameters (for example, if all X_i are binary variables, the number of parameters is 2^n). In terms of computational storage and computing power this is an unfeasible task.

However, the general case assumes that there is not any independence relationship among the variables, which would decrease the storage needs for the

distribution. In fact, a probability distribution modeling some real world phenomena naturally contains many independence relationships. The limit case, a set of totally independent variables (i.e. $P(X_1, \ldots, X_n) = \prod_{i=1}^{n} X_i$), requires only n parameters.

The reduction of the parameter number is not only useful for computational purposes, but to make a better estimation from data. In order to make a more reliable estimation of the probability distribution of the variables, less data is required if the number of independence relationships is larger.

2.2.2 Definition

Given a set of random variables $\mathbf{U} = \{X_1, \ldots, X_n\}$, a *dependency model* M is a set of independence relationships that is enough to determine, for any $\mathbf{A_1}, \mathbf{A_2}, \mathbf{A_3} \subset \mathbf{U}$ (where $\mathbf{A_1}, \mathbf{A_2}, \mathbf{A_3}$ are disjoint, and $\mathbf{A_1}, \mathbf{A_2}$ are not empty) if the independence relationship $\mathbf{A_1} \perp \mathbf{A_2} | \mathbf{A_3}$ is true or not.

If we have a set of random variables arranged (one on each node) in a directed acyclic graph (DAG) [44], we shall use the following notation. If there is a link from a node X to a node Y, we shall say that "X is the *child* of Y" or, alternatively, "X is the *parent* of Y". Extending this notation, we could also define the set of *descendants* of a node (their children, the children of them, and so on), and the *ascendants* (the parents, their parents, and so on). Although formally they are different concepts, in this context "random variable" and "node" are usually interchanged for brevity. Thus, we could be talking of "the set of parents of the variable X" or "the probability distribution of the node X".

The *Markov assumption* says that a random variable of the graph is independent of its non-descendants, given its parents (and only its parents).

A DAG G of random variables is an *I-Map* of a distribution P (on the same set of variables) if all the Markov assumptions implied by G are satisfied by P.

One *I-map* is *minimal* if after the removal of one arc it stops being an I-map.

A probability distribution P on a DAG D forms a *Bayesian network* [14; 57; 58; 102] if and only if D is a minimal I-map of P. Therefore, a Bayesian network over a set of random variables can be seen as a set of independence relationships stored by means of a DAG.

Theorem 2.2.1. (Factorization of a Bayesian Network) *The joint distribution of a Bayesian network $P(X_1, \ldots, X_n)$ factorizes as follows:*

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | Pa(X_i))$$

Where $Pa(X_i)$ is the set of parents of the variable on the graph (none if the node has no parents).

Proof. By the chain rule, $P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_{i-1}, \ldots, X_1)$. Without loss of generality, we can assume that $X_1, \ldots, X_n$ follows an order consistent with that defined by the graph (i.e. $Pa(X_i) \subset \{X_{i-1}, \ldots, X_1\}$ and $\{X_{i-1}, \ldots, X_1\} \setminus Pa(X_i) \subset NonDesc(X_i)$, where $NonDesc$ represents the set of non descendants of a variable).

Since the graph is an I-map, it holds that $X_i \perp NonDesc(X_i) | Pa(X_i)$ by its definition. By the property of decomposition, and the previous assumption, it also holds that $X_i \perp \{X_{i-1}, \ldots, X_1\} \setminus Pa(X_i) | Pa(X_i)$.

Thus, $P(X_i | X_1, \ldots, X_{i-1}) = P(Xi | Pa(X_i))$. □

Surprisingly, the converse result also holds, relating a distribution which factorizes that way with the set of independences represented by a Bayesian network:

Theorem 2.2.2. *Let be a distribution P on a graph such that $P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | Pa(X_i))$. Then, the graph is an I-map of P.*

We shall not prove this theorem here (see, for instance [102] instead).

To sum up, we shall say that a *Bayesian network* is made of two components. The former is a DAG $G = (V, E)$ where the *nodes* (the finite set V) are the random variables concerning the problem we are modeling, and the *edges* ($E \subset V \times V$) indicate dependence relationships among variables. The latter is a set of probability distributions (one for node), representing the conditional probability distribution of the node, conditioned to the possible values of any of the *parents* (the nodes without any parents store the prior probability distribution of their own values).

Bayesian networks verify the set of Markov assumptions, but the concept of *d*-separation (a graphical criterium of independence) is often used instead, because

it seems to be more useful, in order to perform computations. We shall not prove the equivalence between the set of Markov assumptions and the independences obtained by d-separation (for a proof see, for instance, [102]).

2.2.3 Graphical Criteria of Independence

Bayesian networks provide a concise set of *graphical criteria* to check if two sets of random variables are independent, given another set.

We give here some definitions before stating this criterion:

We shall say that a node in a Bayesian network, for two incident arcs on it is a *head-to-head* node, if both arcs point to it. On the contrary, we shall say that the node is *tail-to-head* (see figure 2.1).

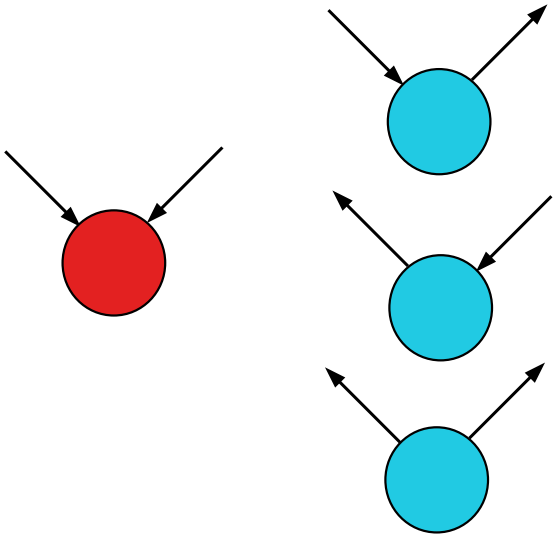

Figure 2.1: On the left side, a head-to-head node. On the right, the three possible configurations for a tail-to-head one.

We shall say that X is d-separated (or just *separated*) of Y given $Z_1, \ldots, Z_k$ if all path (traversing the arcs of the networks in both directions) between X and Y is *blocked* by any node of $Z_1, \ldots, Z_k$. The two possible ways one node blocks a path (d-separation) are:

- An observed tail-to-head node exists on the path.

- A unobserved head-to-head, with all its descendants also unobserved, exists on the path.

After these two definitions we present the main result relating d-separation and independence:

Theorem 2.2.3. *Given a Bayesian network, if A and B are two variables of it, separated by C, then $A \perp B|C$ holds on it.*

An alternative definition of I-map in terms of d-separation is the following. One DAG over a set of random variables is an I-map of a dependency model M if the d-separation over two sets of variables implies that they are independent.

Thus, a Bayesian network of nodes X_i, $i = 1, \ldots, n$ is a useful expression to find probabilities $p(x_1, \ldots, x_n)$ because the set of variables satisfies the list of independences represented by the graph. This graphical expression of the distribution has helped to develop algorithms to apply some tools of Probability Theory (as the Bayes' Theorem, or the marginalization rule) in order to compute probability values (which will be explained in section 2.2.4). Moreover, inferring the probability distribution from data, assuming that there is a Bayesian network representing the set of independences among the variables is also possible, and a very studied problem [11] (in fact, one example of these algorithms will be reviewed on section 2.2.5).

2.2.4 Inference Algorithms for Bayesian Networks

Inference in Bayesian networks [26; 58; 102] is the method by which, given a prior knowledge (called *evidence*), we can discover another one; that is to say, we can compute the probabilities of certain results happening.

Formally, given a set of random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, $\mathbf{E} \subset \mathbf{X}$, being $\mathbf{E} = \{E_1, \ldots, E_m\}$ (*evidential* variables), and a set of values $\mathbf{e} = \{e_1, \ldots, e_m\}$, such that $e_i \in \Omega_{E_i}$, an *inference algorithm* or a *propagation algorithm* is a method that computes the probability value

$$p(x_i|\mathbf{e}), \quad x_i \in \Omega_{X_i}, \quad \forall X_i \in \mathbf{X} \backslash \mathbf{E}.$$

Note that the set of observed variables can be an empty set.

If the probability value $p(x_i|\mathbf{e})$ is computed in an exact manner, the algorithm is called an *exact* propagation algorithm. If we compute an approximation $\widehat{p}(x_i|\mathbf{e})$ to $p(x_i|\mathbf{e})$, the algorithm is called *approximate* [13].

The need for approximate algorithms is due to the computational complexity of inference in Bayesian networks. The problem of exact inference for the general case of Bayesian networks is NP-hard [30]. Although the approximate inference problem is also NP-hard [33], these methods require less computational overhead, and perform relatively well in cases where it is not possible to make exact inference.

Exact inference algorithms [26; 102] are easy to understand, and they are based on the probability theory concepts explained before. Among them we have the *variable elimination algorithm* or the *clique tree propagation* (see, for instance [102]).

Some examples of approximate inference algorithms are the *Monte Carlo methods* [13], among which we have the *importance sampling algorithm* [84]. Other approaches are *likelihood weighting* or *probabilistic logic sampling* (see, for instance [99]).

2.2.5 Learning algorithms for Bayesian Networks

2.2.5.1 Concept

The task of *learning* a Bayesian network consist of recovering the graph structure and the set of probability distributions, provided a set of samples from the joint distribution.

Formally, if we get a set $(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)})$ of samples from the set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, our objective is to find the Bayesian network which best represents the data. In order to find an "optimal" solution, we must state that the problem comprises two different steps:

1. The learning of the *structure* of the Bayesian networks.

2. The learning of the set of *parameters*.

The second problem has been very well studied, and has an exact solution (see, for instance, [99]).

The first problem can be solved using two different approaches:

1. Detecting the set of independences among the variables (which leads to get the Bayesian network structure).

2. Searching for a network which represents correctly the samples we get.

The first scheme detects independences by making some tests, and adding the detected independences to a list. Then, a Bayesian network which represents the majority (or all) of these independences is given as an output.

The second scheme (called *search and score*) searches in the space of all possible Bayesian network structures, assigning to each of it one a real value (score) computed with a certain function (metric), which measures the adequacy of the network to the data.

In both cases, the search space (the set of all possible directed acyclic graphs among n variables) is of *hyper exponential* size, which gives an idea of the intractability of the problem. In fact, the search for the optimal graph structure using a score is not computationally affordable. In the work by Chickering, Geiger and Heckerman [29] the first result on the complexity of learning Bayesian networks is presented. That result is stated as follows: the algorithm for learning networks with, at most k parents ($k > 1$[1]), having a posterior probability greater than a certain constant, using the BDe metric is NP-hard.

In order to find a good network structure, approximate search algorithms are used. Among them we can find lot of heuristic and metaheuristics approaches which usually find good solutions (although they are probably local optimum). A very simple approach to search for a network structure, using the easiest heuristic search algorithm is the hill climbing algorithm, presented in the following section.

[1]The algorithm for learning optimal networks where the graph is a *tree* ($k = 1$) is polynomial, and perfectly known [102].

2.2.5.2 The Hill Climbing Algorithm

This procedure is an application of the "classic" heuristic search algorithm. This algorithm starts with an initial solution to this problem (for example the "empty" network, i.e. a network whose graph has no arcs). To perform hill climbing, a *neighbor operator* is needed. A neighbor of a solution to the problem is a point in the search space which is close to the current solution (that is to say, we can obtain a neighbor from the solution by doing minimal perturbations on it).

For the case of Bayesian networks, we can consider that a neighbor of one network is the same network, either with a new arc added[1], or the initial network, with one of its arcs removed. We could consider other different neighbors, but these two are enough for our purposes. Our operator, then, would generate all the different neighbors of the current solution. See figure 2.2 for an example of the neighbors of a small network.

The algorithm will choose the neighbor which makes the best improvement of the score, compared to the current solution. That is to say, we "climb the hill following the steeped path". If at any moment, we cannot find a neighbor which improves our solution, we shall stop our algorithm and return the current solution.

The skeleton of this procedure is represented on algorithm 1.

2.3 Canonical Models

2.3.1 Introduction

Building probabilistic graphical models, like Bayesian networks, requires the specification of a large number of parameters. Concretely, for each node Y in the network, we shall state the probability distributions $P(Y = y|Pa(Y))$, where, one more time, $Pa(Y)$ represents the set of parent nodes of Y.

For the specific case of binary variables $Y = \{y, \bar{y}\}$, and $|Pa(Y)| = n$, the number of parameters to estimate is $2^n - 1$. Therefore, we are interested in keeping a small number n of parents in order to perform good estimations. In certain

[1] Note that the arc should be added avoiding cycles.

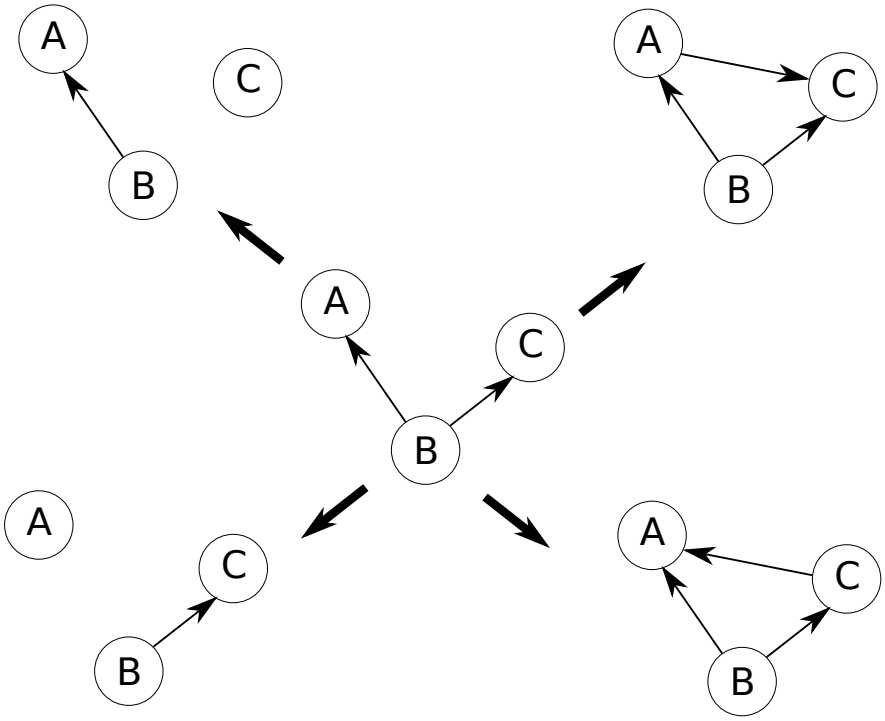

Figure 2.2: Set of neighbors (on the corners) for a given Bayesian network (on the center).

domains, the previous requirement is not very easy to assume. In the treatment of text documents with Bayesian networks (as we shall see in subsequent chapters) you can find a node with hundred or thousand of parents. A reasonable solution to this problem is given using *canonical models* [42].

These models permit, after several assumptions, building Bayesian networks where the nodes have lot of parents, avoiding the estimation of an exponential number of parameters on the number of variables (in fact, they often require only to estimate a number of parameters which is a linear function on the number of parents).

Informally, building a canonical model over a node Y, with parents $\mathbf{X} = \{X_1, \ldots, X_{|Pa(Y)|}\}$ implies having a method to compute efficiently the probability values $p(y|\mathbf{x})$, where $\mathbf{x}$ is any configuration among all the possible in the set $\mathbf{X}$ of parents of Y.

We shall recall two models in this section: the *noisy OR gate* model, and the

Algorithm 1 Hill climbing algorithm for learning a Bayesian network structure

```
Input: metric, set of samples S = (x^(1), ..., x^(m))
currentNetwork ← empty Bayesian network
loop
  L ← NEIGHBORS(currentNetwork)
  nextEval ← −∞
  nextNetwork ← ∅
  for all candidate ∈ L do
    if (metric(S, candidate) > nextEval) then
      nextNetwork ← candidate
      nextEval ← metric(S, candidate)
    end if
  end for
  if nextEval ≤ metric(S, currentNetwork) then
    /* Return current network since no better neighbors exist */
    return currentNetwork
  end if
  currentNetwork ← nextNetwork
end loop
```

additive model, which will be used in subsequent chapters.

2.3.2 Noisy OR Gate Model

The noisy OR gate model [57; 102] is a causal model which represents *disjunctive* interaction among the different causes of one effect (again, the causes are the parents of the variable). We shall assume that both the causes ($\mathbf{X} = \{X_1, \ldots, X_{|Pa(Y)|}\}$) and the effect ($Y$) are binary variables (i.e. they can occur or not).

This interaction occurs if any of the causes can produce the effect in an independent manner, and with no possibility of an inhibition among them if more than one are present at the same time. We need two additional conditions to use this kind of model.

- The effect event is false (its probability is zero) if all of its causes are set to false.

- If an event is a consequence of two conditions C_1 and C_2, then, the mechanism that inhibits the occurrence of that event with the occurrence of the condition C_1, is independent of the mechanism that inhibits the occurrence of that event when the condition C_2 is given.

We show the formula for the probability that the effect Y occurs, given its parents X_i:

$$\forall X_i \in \mathbf{X}, \quad p(y|\mathbf{X}) = 1 - \prod_{X_i \in R(\mathbf{x})} (1 - w(X_i, Y)) \,.$$

Here, $R(\mathbf{x})$ represents the set of variables of the current configuration of $\mathbf{X}$ which have been observed. Obviously $p(\overline{y}|\mathbf{X}) = 1 - p(y|\mathbf{X})$.

In the previous formula $w(X_i, Y)$ is a positive weight, between 0 and 1, representing the probability that the effect Y occurs, being true the cause X_i, and false all $X_j, j \neq i$ (from a certain point of view, a measure of the "causality" among cause and effect).

If the weights $w(X_i, Y)$ are all equal to 1, the model acts like a *pure OR gate* (any of the causes can make the effect happen, with probability 1, with its only occurrence).

2.3.3 The Additive Model

The additive model is a canonical causal model, used on Information Retrieval [15], and it represents an interaction among variables, where the following properties hold:

- The occurrence of an additional cause (apart from those which are known) never lowers the likelihood of the occurrence of the effect.
- The probability of occurrence of the effect will be maximum if all the causes occur.
- All the causes are independent.

Analytically, we can represent the probability that Y occurs as:

$$\forall X_i \in \mathbf{X}, \quad p(y|\mathbf{X}) = \sum_{X_i \in R(\mathbf{x})} w(X_i, Y)\,, \tag{2.2}$$

where the value $w(X_i, Y)$ represents the importance (weight) associated to each variable X_i over the effect Y. Again, $R(\mathbf{x})$ also represents the set of variables of the current configuration of $\mathbf{X}$ which have been observed. Thus, if more parents are observed, the probability of $Y = y$ will be higher.

Obviously, we should add a restriction on the weight values: they can be defined in any way, satisfying

$$\forall i = 1, \ldots, |Pa(Y)|, \quad w(X_i, Y) \geq 0,$$

and

$$\sum_{X_i \in Pa(Y)} w(X_i, Y) \leq 1.$$

If we have no reason to believe than one parent is more important than the others, we can simply define $w(X_i, Y) = \frac{1}{|Pa(Y)|}$, which is equivalent of applying the Keynes principle of indifference [99].

If we have a certain prior information about the importance of a parent, namely $f(X_i)$, we can define the weights by normalizing that importance between 0 and 1, dividing by the total sum of importances. Thus:

$$w(X_i, Y) = \frac{f(X_i)}{\sum_j f(X_j)}$$

Those two ways to define weights satisfy the previous weight requirements, and will be used later, in chapter 4.

Part II

Methodological Contributions & Applications

Chapter 3

An OR Gate-Based Text Classifier

3.1 Introduction

This chapter is devoted to the construction of several text classifiers based on Bayesian networks, concretely on the model known as the *noisy OR gate*. This is a very well known model, and with a wide usage on knowledge engineering [43].

We shall organize this chapter as follows. The use of this kind of probabilistic models (noisy OR gates) should be well motivated, so we shall start with a section (in 3.2) which presents the advantages of using this kind of classifier on this domain, and shows the need of doing more research in basic Text Categorization. Given that there exists two different modalities of probabilistic classification –generative and discriminative–, we present them in section 3.3, explaining these two approaches. In section 3.4 we list some related works (usages of the noisy OR gate in Text Classification) which result in being alternatives to our model. After the prelude part, in section 3.5 we shall describe the proposed new model. In fact, while being developed, we shall describe two different parameter estimation, which will end giving two different models.

In order to fix the potential drawbacks of the models, and before doing experimentation, we shall present several improvements in both of them, adding a pruning procedure, which can be seen as a refinement of the training algorithm.

This will result in two new versions of the presented classifiers, which will be detailed on section 3.6.

Finally, section 3.7 is focused on the experimental results, testing the presented models on some standard corpus, and section 3.8 contains the concluding remarks and several proposals for future work, ending the chapter.

3.2 Motivation

Prior to describe the models, which is the objective of the chapter, this section tries to answer the two following questions that one can ask when reviewing our work in this area:

- *Why doing some research on basic Text Categorization?*
- *Why are we interested in using a model based on noisy OR gates?*

The two following subsections are entitled with each one of the two previous questions, respectively, trying to find a reasonable answer for each one of them.

3.2.1 Why doing some research on basic Text Categorization?

This question is motivated by the current state of the art in Text Categorization. As we said in chapter 1, the results obtained by methods like support vector machines (even in the linear case, the simplest one) are always on the top of the list of best performing models on the standard testing corpus [46]. Other methods like, for instance, the logistic regression classifier, tend also to be on high places on the ranking of models. Therefore, in both cases, the numeric results obtained by these models are impressive (the linear support vector machine, for example, reaches a $F_1 = 0.92$ for some categories of Reuters) which are nearly as perfect as an expert human indexer performing that task.

Then, what is the point in developing new models from scratch? The main drawbacks of linear support vector machines are the following:

- The learning procedure is an *iterative method* which minimizes the empirical risk. Although this estimation can be seen as a special case of Bayesian decision theory [91], this explanation is found a posteriori, and the basic method only tries to find a "good" separating hyperplane on the training data, without looking at the distribution of the data. That is to say, the algorithm is like a "black box" that makes a *regression* on the training data, finding an optimum vector. In fact, those vector coordinates are not an obvious function on individual term frequencies.

- Besides, because the procedure relies on an optimization algorithm, it might be of a *long execution time* for certain problems (the case of Text Categorization, for example, where both the number of variables and instances is high, and vectors should be read from disk).

- Although they rely on a very strong theory (Statistical Learning Theory), learning algorithms are *difficult to develop*.

- The set of outputs given from this algorithm are *not easily combinable* with other classifiers, because they can be any real number. Although Platt [103] proposed fitting a logistic function to the output of the classifier, this is not its original formula (and this procedure is highly expensive in time, because it requires training several classifiers on the training set via cross validation to tune the thresholds).

- The learnt function is almost *impossible to update*, in case that new training data were made available.

It occurs the same for logistic regression [50], where a real vector is fitted to the data, in order to maximize the likelihood of the model, given the data, using some classic optimization procedure. It is a black-box, long time consuming, and difficult to develop algorithm. For this case, its output is probabilistic, which makes it more interpretable (and easier to use it, in posterior stages, or combining with other classifiers).

On the other hand, there are simpler methods as the Naïve Bayes, which tend to perform reasonably well [85] on complex environment. Unjustly called "the

punching bag of classifiers", it has been shown that, with few changes on the algorithms, and keeping its simplicity, a simple Naïve Bayes can reach the same level that a linear support vector machine [106]. The Naïve Bayes has a very simple and fast training procedure, is easy to update if new data is available, its output is a probability, and it is widely used on real applications. Its training procedure relies on the fact that the terms in the documents are assumed to follow any particular statistical distribution (as, for example multinomial or Bernoulli distributions) and so, some estimation is carried out (which results on the training algorithm).

We would like to prospect for the same direction given by the Naïve Bayes. That is to say, *simple algorithms*: where the training is reduced to an estimation procedure, and for testing it only requires to compute a posterior distribution, with very few parameters. We think that this line is not exhausted for the problem of Text Categorization, being almost all the modern results coped by statistical machine learning, but few simple algorithms. This is our main reason for doing this kind of "basic" research, on a field that, as a first sight (and simplistic), seems to be run out [118].

3.2.2 Why are we interested in using a model based on noisy OR gates?

There are several answers to this question. To begin with, the noisy OR gate is part of a bigger family of probabilistic models called the "models of multicausal interaction" [102] or simply "canonical models" [42]. The main aim of these models is to represent, in a more compact way, probability distributions which, in the general case, would be impossible to estimate (and store). For example, a noisy OR gate of n parents with binary variables can be represented with only n parameters, opposite to the $2^n - 1$ parameters of the general case. In a few words, canonical models come to approximate general Bayesian networks, for the case of nodes with many parents.

The choice of one of those models in real tasks is motivated by some previous assumptions which have to be taken, in order to use one of these models with data. If the assumption is reasonable for our data, and the model represents the

interactions among variables we want to modelize, the canonical model can be a good candidate for some tasks.

For the particular case of noisy OR gates, they are models which have acquired great success in knowledge representation [43], particularly on the fields of medical problems, where its simplicity has avoided estimating a large number of parameters. Dealing with text documents is linked with using a huge number of variables, and all the possibilities to reduce the number of estimations are surely welcomed.

For the general case of canonical models, and some related applications of IR, canonical SUM models (like those presented on chapter 2) have been used with great success in tasks of classic Information Retrieval [15] and Structured Information Retrieval [32]. We think that both the application (Document Classification) we are working on, and the method (noisy OR gates) we are using, are naturally close to these works.

Finally, we shall give a more objective reason to use noisy OR gates. If we think in the simplest probabilistic model, the Naïve Bayes, it is known that it relies on the conditional independence of the terms, given the category (the so-called Naïve Bayes hypothesis). In order to overcome this simplification, several methods have been proposed to make classifiers where higher than one order dependences are taken. In fact, Sahami [113] proposed the limited dependence classifier which lies between the Naïve Bayes and the general Bayesian network classifier (a classifier where the network is inverted with respect to the Naïve Bayes one). Note that this general network assumes that all the terms are instantiated (to positive or negative), and therefore, no relationships among the terms are needed.

In figure 3.1 we can see those three dependence models: the Naïve Bayes (which can be stated as a 1-dependence model) labeled with (a), a generic k-dependence model ($k > 1$, where each term variable has, at most, k parents) labeled with (b), and finally the general Bayesian network classifier, a hypothetically better classifier, where the dependences among the terms and the category variables would be better estimated (all the terms are dependent among them, given the category variable). If we think in the complete Bayesian network as the ideal model of classification, it is perfectly clear that we do not have enough data

available to make estimators, and the number of required parameters would be very high (exponential on the number of variables). So, our way to approximate this classifier is training a discriminative canonical model on the data, which, at least will try to represent the same set of independences than the Bayesian network classifier.

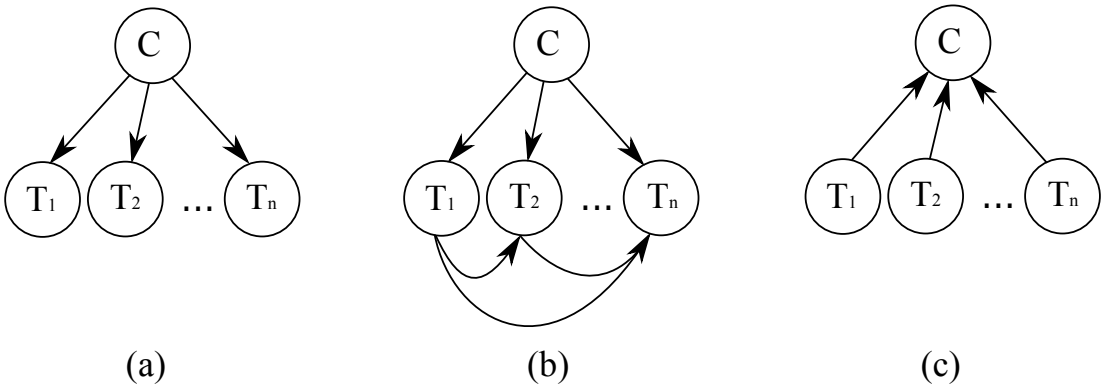

Figure 3.1: Three possible network structures for probabilistic classifiers.

3.3 Generative and Discriminative Methods

As we explained in chapter 2, a very popular approach to text categorization are probabilistic classifiers. Among them, we can find several versions of Naïve Bayes classifiers [64; 76; 85; 106], the limited dependence classifiers [113] or the family of logistic regression-based classifiers [50].

The classical approach to probabilistic text classification may be stated as follows: we have a class variable C taking values in the set $\{c_1, c_2, \ldots, c_n\}$[1] and, given a document d_j to be classified (described by a set of attribute variables, which usually are the terms appearing in the document), the posterior probability

[1]Recall that the simplest case is $C = \{c, \overline{c}\}$ (the *binary* case, where we choose between a category and its opposite), and when the number of values is greater than 2, we deal with a *multiclass* problem. If the number of assigned labels to each document is greater than 1, we have a *multilabel* problem, and we often reduce it to a set of *binary* problems, with one class variable $C_i = \{c_i, \overline{c}\}$ for each label. In that case, each probability distribution is estimated independently and will be noted as p_i, to emphasize this fact. When discussing general cases, like this section, we shall deliberately skip the i subindex, to avoid confusion.

of each class, $p(c_i|d_j)$, is computed in some way, and the document is assigned to the class having the greatest posterior probability. Learning methods for probabilistic classifiers are often characterized as being *generative* or *discriminative*.

Generative methods estimate the joint probability distribution of all the variables, $p(c_i, d_j)$, and therefore $p(c_i|d_j)$ is computed according to the Bayes' rule:

$$p(c_i|d_j) = \frac{p(c_i, d_j)}{p(d_j)} = \frac{p(c_i)\, p(d_j|c_i)}{p(d_j)} \propto p(c_i)\, p(d_j|c_i)\,. \tag{3.1}$$

The problem in this case is how to estimate the probabilities $p(c_i)$ and $p(d_j|c_i)$. In contrast, discriminative probabilistic classifiers model the posterior probabilities $p(c_i|d_j)$ directly.

The Naïve Bayes classifier is the simplest generative probabilistic classification model that, despite its strong and often unrealistic assumptions, performs frequently surprisingly well. It assumes that all the attribute variables are conditionally independent on each other given the class variable. In fact, the Naïve Bayes classifier can be considered as a Bayesian network-based classifier [1], where the network structure is fixed and contains only arcs from the class variable to the attribute variables, as shown in figure 3.2.

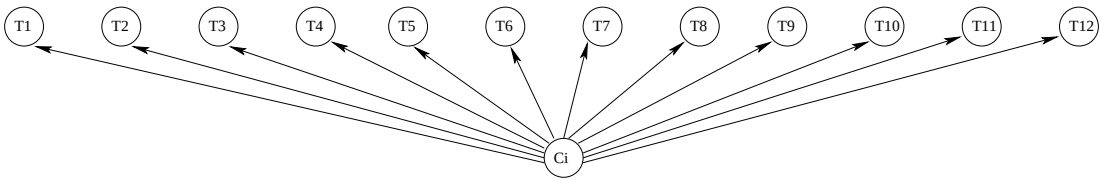

Figure 3.2: Network structure of the Naïve Bayes classifier.

Here, we shall propose another simple Bayesian network-based classifier, which may be considered as a discriminative counterpart of Naïve Bayes (presented in chapter 1) in the following senses:

1. it is based on a type of Bayesian network similar to the Naïve Bayes one, but with the arcs in the network going in the opposite direction;

2. it requires the same set of simple sufficient statistics than Naïve Bayes, so that the complexity of the training step in both methods is the same; the complexity of the classification step is also identical.

If we assume the list of variables $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$ for the set of terms, the objective of probabilistic classification is to estimate the probability distribution $P(C|T_1, T_2, \ldots, T_N)$.

3.4 Related work

In this section we briefly review other approaches to text categorization based on noisy OR gates, or in similar models.

In [129], a classifier very similar to this is presented. A noisy OR gate with one class variable as effect, and term as causes. In this case, the training procedure is completely different than our approach, using the EM algorithm [35] to find the set of parameters that maximize the likelihood of the data. The model is tested on the Reuters collection, with notable results compared to linear support vector machines.

In [63] the previous model of [129] is presented on a more general framework (what they name the set of *symmetric causal independence models*). Moreover, they show that the maximum found by [129] is a global one (due to the concavity of the likelihood function). Some other related models are also presented, and a set of experiments with the Reuters dataset and others are also carried out with good results.

The model presented here was firstly shown by us on the field of Structured (XML) Document Categorization (the problem presented in chapter 5), in [16], although its results in flat document categorization were better than in transformed documents.

3.5 The OR Gate Bayesian Network Classifier

The document classification method that we are going to propose, presented before in [18], is based on another restricted type of Bayesian network with the following topology: each term t_k appearing in the training documents (or a subset of these terms in the case of using some method for feature selection) is associated to a binary variable T_k taking its values in the set $\{t_k, \bar{t}_k\}$, which in turn is represented in the network by the corresponding node. There are also n binary

variables C_i taking its values in the sets $\{c_i, \overline{c}_i\}$ (as in the previous binary version of the Naïve Bayes model) and the corresponding class nodes. The network structure is fixed, having an arc going from each term node T_k to the class node C_i if the term t_k appears in training documents which are of class c_i. Let nt_i be the number of different terms appearing in documents of class c_i. In this way we have a network topology with two layers, where the term nodes are the "causes" and the class nodes are the "effects", having a total of $\sum_{i=1}^{n} nt_i$ arcs. An example of this network topology is displayed in figure 3.3. It should be noticed that the proposed topology, with arcs going from attribute nodes to class nodes, is the opposite of the one associated to the Naïve Bayes model.

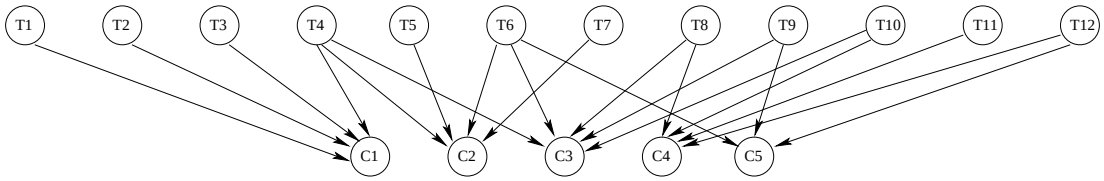

Figure 3.3: The OR gate classifier network structure.

It should also be noticed that this network topology explicitly requires modeling the "discriminative" conditional probabilities $p_i(c_i|pa(C_i))$, where $Pa(C_i)$ is the set of parents of node C_i in the network (the set of terms appearing in documents of class c_i) and $pa(C_i)$ is any configuration of the parent set (any assignment of values to the variables in this set). We have said in section 3.2 that, as the number of configurations is exponential with the size of the parent set[1], we use a canonical model to define these probabilities, which reduces the number of required numerical values from exponential to linear size. More precisely, we use a noisy OR Gate model [102].

Recall that (see chapter 2) the conditional probabilities in a noisy OR gate are defined in the following way:

$$p_i(c_i|pa(C_i)) = 1 - \prod_{T_k \in R(pa(C_i))} (1 - w(T_k, C_i)) \,, \tag{3.2}$$

$$p_i(\overline{c}_i|pa(C_i)) = 1 - p_i(c_i|pa(C_i)) \,, \tag{3.3}$$

[1]Notice that $|Pa(C_i)| = nt_i$.

where $R(pa(C_i)) = \{T_k \in Pa(C_i) \,|\, t_k \in pa(C_i)\}$, i.e. $R(pa(C_i))$ is the subset of parents of C_i which are instantiated to its t_k value in the configuration $pa(C_i)$. $w(T_k, C_i)$ is a weight representing the probability that the occurrence of the "cause" T_k alone (T_k being instantiated to t_k and all the other parents T_h instantiated to $\bar{t}_h$) makes the "effect" true (i.e., forces class c_i to occur).

Note that the definition of the probability distributions is independent on the kind of problem considered. We will define n binary distributions for C_i variables, even for a multiclass problem. As we shall see in next subsection, this is not a drawback to work in this kind of problems.

3.5.1 Classification as Inference

Once the weights $w(T_k, C_i)$ have been estimated, and given a document d_j to be classified, we instantiate in the network each of the variables T_k corresponding to the terms appearing in d_j to the value t_k (i.e. $p(t_k|d_j) = 1$ if $t_k \in d_j$), and all the other variables T_h (those associated to terms that do not appear in d_j) to the value $\bar{t}_h$ (i.e. $p(t_h|d_j) = 0 \ \forall t_h \notin d_j$). Then, we compute for each class node C_i the posterior probabilities $p_i(c_i|d_j)$. As in the case of the Naïve Bayes model, we would assign to d_j the class (or classes) having the greatest posterior probability.

The combination of network topology and numerical values represented by OR gates allows us to compute very efficiently and in an exact way the posterior probabilities:

$$\begin{aligned} p_i(c_i|d_j) &= 1 - \prod_{T_k \in Pa(C_i)} (1 - w(T_k, C_i) p(t_k|d_j)) \\ &= 1 - \prod_{T_k \in Pa(C_i) \cap d_j} (1 - w(T_k, C_i)) \,. \end{aligned} \tag{3.4}$$

In order to take into account the number of times a word t_k occurs in a document d_j, n_{jk}, we can replicate each node T_k n_{jk} times, so that the posterior probabilities then become:

$$p_i(c_i|d_j) = 1 - \prod_{T_k \in Pa(C_i) \cap d_j} (1 - w(T_k, C_i))^{n_{jk}} \,. \tag{3.5}$$

Note that, after categorization, we have a set of values $p_i(c_i|d_j)$ representing the CSV for the category i and the document j. If we want to make multiclass classification, we can use the following classification rule: choose the category C_l such that,

$$l = \arg \max_{k \in \{1,2,\ldots,|\mathcal{C}|\}} p_k(c_k|d_j),$$

or, in other words, choose the category with greatest posterior probability.

3.5.2 Training as Weight Estimation

The estimation of the weights in the OR gates, $w(T_k, C_i)$, can be done in several ways. The simplest one is to compute $w(T_k, C_i)$ as $\hat{p}(c_i|t_k)$, the estimated conditional probability of class c_i given that the term t_k is present. We can do it by maximum likelihood:

$$w(T_k, C_i) = \frac{N_{ik}}{N_{\bullet k}}, \tag{3.6}$$

or using Laplace correction:

$$w(T_k, C_i) = \frac{N_{ik} + 1}{N_{\bullet k} + 2}. \tag{3.7}$$

Recall (see chapter 1) that N_{ik} is the number of times that t_k appears in documents of class c_i, $N_{\bullet k}$ is the numbers of times that the term t_k appears in the training documents, i.e. $N_{\bullet k} = \sum_{c_i} N_{ik}$. N is the total number of words in the training documents.

Another, more accurate way of estimating $w(T_k, C_i)$ is directly as $\hat{p}_i(c_i|t_k, \bar{t}_h \, \forall T_h \in Pa(C_i), T_h \neq T_k)$. However, this probability cannot be reliably estimated, so that we are going to compute an approximation in the following way:

$$\hat{p}_i(c_i|t_k, \bar{t}_h \, \forall h \neq k) = p_i(c_i|t_k) \prod_{h \neq k} \frac{p_i(c_i|\bar{t}_h)}{p_i(c_i)}. \tag{3.8}$$

This approximation results from assuming a conditional independence state-

ment similar to that of the Naïve Bayes classifier, namely

$$p_i(t_k, \bar{t}_h \, \forall h \neq k | c_i) = p_i(t_k|c_i) \prod_{h \neq k} p_i(\bar{t}_h|c_i). \tag{3.9}$$

In that case

$$\begin{aligned} p_i(c_i|t_k, \bar{t}_h \, \forall h \neq k) &= \frac{p_i(t_k, \bar{t}_h \, \forall h \neq k|c_i) p_i(c_i)}{p_i(t_k, \bar{t}_h \, \forall h \neq k)} \\ &= \frac{p_i(t_k|c_i) \left(\prod_{h \neq k} p_i(\bar{t}_h|c_i) \right) p_i(c_i)}{p_i(t_k) \prod_{h \neq k} p_i(\bar{t}_h \, \forall h \neq k)} \\ &= p_i(c_i|t_k) \prod_{h \neq k} \frac{p_i(c_i|\bar{t}_h)}{p_i(c_i)}. \end{aligned}$$

The values of $p_i(c_i|t_k)$ and $p_i(c_i|\bar{t}_h)/p_i(c_i)$ in equation (3.8) are also estimated using maximum likelihood. Then, the weights $w(T_k, C_i)$ are in this case:

$$w(T_k, C_i) = \frac{N_{ik}}{N_{\bullet k}} \prod_{h \neq k} \frac{(N_{i\bullet} - N_{ih})N}{(N - N_{\bullet h})N_{i\bullet}}. \tag{3.10}$$

Another option is to relax the independence assumption, consider interactions among the terms, in the following way:

$$p_i(t_k, \bar{t}_h \, \forall h \neq k | c_i) = \frac{p_i(t_k|c_i)}{nt_i} \prod_{h \neq k} p_i(\bar{t}_h|c_i) \, . \tag{3.11}$$

We are assuming that the joint probability of the events $\{t_k, \bar{t}_h \, \forall h \neq k\}$ is smaller than the pure independence assumption would dictate. The weights $w(T_k, C_i)$ would be in this case

$$w(T_k, C_i) = \frac{N_{ik}}{nt_i N_{\bullet k}} \prod_{h \neq k} \frac{(N_{i\bullet} - N_{ih})N}{(N - N_{\bullet h})N_{i\bullet}}. \tag{3.12}$$

In any case, the set of sufficient statistics necessary to compute the weights are $N_{ik} \; \forall t_k, \forall c_i$, i.e. the number of times each term t_k appears in documents of each class c_i, the same required by multinomial Naïve Bayes.

Some preliminary experimentation showed that the best performing methods were those based on equation 3.7 (for estimation based on $\hat{p}_i(c_i|t_k)$) and equation 3.12 (for estimation based on $\hat{p}_i(c_i|t_k, \bar{t}_h \, \forall h \neq k)$), and so, they will be chosen as our two different OR gate models. From this point, and for simplicity notation, we will use this convention to name our two different proposed models:

- The noisy OR gate model, where the weights $w(T_k, C_i)$ are estimated following equation (3.7), will be called the *OR gate ML model*, or simply *OR ML* (where ML stands for "maximum likelihood").
- The noisy OR gate model, where the the weights $w(T_k, C_i)$ are estimated following equation (3.12), will be called the *OR gate TI model*, or just *OR TI* (where TI stands for "term interaction").

3.5.3 A brief note on scaling probability results for OR gate models in multilabel problems

When doing binary or multiclass categorization, the decision rule is to choose the category with greater posterior probability. For the case of multilabel, the problem gets harder.

If we want to make "hard categorization", for probabilistic models which are trained maximizing the likelihood of the data, one possibility is to choose classifying a document d_j on category c_k if $p_k(c_k|d_j) > 0.5$, and not classifying it by c_k otherwise. Here, the decision rule is associated to a threshold, which, for this case is 0.5 (because the parameters has been trained in that way).

The case with our models is different. To begin with, probability values $p_i(c_i|d_j)$ are not directly comparable among different values of i and d_j, because they seem to be on a different scale. One of the reasons for this problem is that different category variables have also a different number of parents, and then, computations are not normalized for each node.

In case that one wanted to use this model for multilabel categorization, we give two solutions to overcome this problem:

- The easier solution, which has a heuristic justification explained here. The output probability values are normalized, dividing by the probability value

of the most probable category. That is to say, the final probability values $p(c_i|d_j)$ are computed the following way:

$$p(c_i|d_j) = \frac{p_i(c_i|d_j)}{\max_{c_k}\{p_k(c_k|d_j)\}}.$$

This results that the more probable category gets automatically a probability of 1, which may not be very realistic. In fact, it can introduce some unwanted false positives on the zone of assignments of higher probability. Anyway, our experience says that this heuristic produces reasonably good results if we want to make "soft categorization" and compute per category breakeven measures. We will refer to it as "the max heuristic".

- If we want to make "hard categorization", the previous decision rule does not work well, even with the max heuristic. For example, in some problems there are categories which always obtain high probability values (because they have a large number of parents), and then, they always get values very close to 1 $(0.9, 0.85, \dots)$ with previous normalization. In fact, if we set the threshold in smaller values like 0.5, we get a large number of false positives, so it is clear that the threshold should be set to a high value. It is obvious that smaller categories would verify the opposite (they will need a small threshold). For that case, an optimum threshold $\tau_i \in [0, 1]$ can be trained for each category, via cross validation on the training set, and after that, it can be applied to the results on the test set, either assigning c_k if $p_k(c_k|d_j) > \tau_k$, or scaling the different $p_k(c_k|d_j)$ values with different linear continuous piecewise functions which verify

$$f_k(\tau_k) = 0.5,\ f_k(0) = 0,\ f_k(1) = 1\ \forall k = 1, 2, \dots, |\mathcal{C}|.$$

This possibility is explored on chapter 5, section 5.13.2.2, where more details of its properties are explained. Using this "scaling heuristic", and finding a set of reasonably adjusted set of thresholds is the only possibility if we want to make hard categorization. After the scale, we can use the previous decision rule (assign c_k to d_j if $p_k(c_k|d_j) > 0.5$).

3.6 Improving the Models Pruning Independent Terms

As we explained on chapter 2, there are several assumptions that should verify our data, in order to apply a noisy OR gate model. Among them, the causes (terms in our model) should be independent among them, and one cause could make the effect (the category is assigned to the document) by itself.

The independence assumption among terms is not real, but it is similar to the Naïve Bayes assumption. Almost every text categorization model works using the prior fact that terms are considered independent. The second assumption is not so clear. In fact, there are lot of terms which are not giving any information about the category (because its distribution is uniform for all categories), and terms which do not give evidence about a certain category, because its occurrence in this category is almost negligible. On the other hand, the presence of certain terms give "negative" information (the likelihood of being in other category is augmented), but this fact is not very well captured by the noisy OR gate model.

In order to improve our models, we propose a simple pruning procedure which can be summed up as follows: *after estimating the parameters, for each category remove as parents the terms which are independent with the category variable.* This procedure guarantees that the remaining terms have a more likely "causal link" with the category (though it could be on the opposite direction).

To assess if a pair term/category (T_k, C_i) are independent, we propose using a simple independence test, based on the χ^2 statistic. We first compute the mutual information on the term and category:

$$I(T_k; C_i) = \sum_{t \in \{t_k, \bar{t}_k\}} \sum_{c \in \{c_i, \bar{c}_i\}} p(t, c) \log \frac{p(t, c)}{p(t)p(c)}$$

that, coherently with previous notation results in:

$$
\begin{aligned}
I(T_k; C_i) \;=\; & \frac{N_{ik}}{N} \log\left(\frac{N_{ik}\, N}{N_{\bullet k}\, N_{i\bullet}}\right) + \frac{N_{\bullet k} - N_{ik}}{N} \log\left(\frac{(N_{\bullet k} - N_{ik})\, N}{N_{\bullet k}\, (N - N_{i\bullet})}\right) + \\
& \frac{N_{i\bullet} - N_{ik}}{N} \log\left(\frac{(N_{i\bullet} - N_{ik})\, N}{(N - N_{\bullet k})\, N_{i\bullet}}\right) + \\
& \frac{N - N_{\bullet k} - N_{i\bullet} + N_{ik}}{N} \log\left(\frac{(N - N_{\bullet k} - N_{i\bullet} + N_{ik})\, N}{(N - N_{\bullet k})\, (N - N_{i\bullet})}\right)
\end{aligned}
\tag{3.13}
$$

The independence test is based on a proposition by Kullback [69] and can be reformulated as follows:

Theorem 3.6.1. (Kullback) *If X and Y are independent random variables (with nx and ny values, respectively), with $\{(X_1, Y_1), (X_2, Y_2), \ldots (X_N, Y_N)\}$ a set of samples, then $2N\, I(X;Y) \sim \chi^2_{(nx-1)(ny-1),1-\alpha}$ holds for $N \longrightarrow \infty$, at a significance level of α.*

The pruning algorithm is then very simple:

Algorithm 2 Pruning algorithm for our noisy OR gate models

```
Input: α, significance value
for all Category C_i ∈ 𝒞 do
  Train an OR gate model on C_i
  for all Term T_j ∈ C_i do
    if 2N I(T_j, C_i) > χ²_{1,1−α} then
      Keep T_j in Pa(C_i)
      Renormalize w(T_j, C_i), if needed
    else
      Remove T_j from Pa(C_i)
    end if
  end for
end for
```

The renormalization step of $w(T_j, C_i)$ is needed for the OR TI model, because the number of parents nt_i of the category node C_i could have changed. If the number of parents nt_i after the selection has changed to nt'_i, then the weights can

be easily renormalized to $w'(T_j, C_i)$ by doing

$$w'(T_j, C_i) = \frac{nt_i}{nt'_i} w(T_j, C_i).$$

Therefore, this procedure gives us two new classification algorithms, both with a free parameter (α, the level of significance of the χ^2 test). They will be noted as OR ML-α and OR TI-α (the pruned versions of the models presented before, with a certain α value.

3.7 Experimentation

For the evaluation of the proposed model we have used three document test collections: Reuters-21578, Ohsumed and 20 Newsgroups. Reuters-21578 (ModApte split) contains 12,902 documents (9603 for training and 3299 for testing) and 90 categories (with at least 1 training and 1 test documents). Ohsumed, including 20000 medical abstracts from the MeSH categories (10000 for training and 10000 for testing) of the year 1991, and 23 categories. 20 Newsgroups corpus contains 19997 articles for 20 categories taken from the Usenet newsgroups collection, where only the subject and the body of each message were used. Note that there is no fixed literature split for this collection. All the three collections were preprocessed in the same way using stemming (Porter's algorithm) and stopword removal (SMART's system 571 stopword list). No term selection was carried out.

The evaluation takes into account that the classification process will generate an ordered list of possible categories, in decreasing order of probability[1], instead of a definite assignment of categories to each document. Then, as performance measures, we have firstly selected the typical measures used in multi-label categorization problems (as they are Reuters-21578 and Ohsumed): *breakeven* point[2] (BEP) and the *average 11-point precision*[3] (Av-11). Another measure commonly used is F_1[4]. However F_1 requires a precise assignment of classes to each docu-

[1] We are therefore using an instance of the so-called category-ranking classifiers [119].

[2] The point where precision equals recall, by moving a threshold.

[3] The precision values are interpolated at 11 points at which the recall values are 0.0, 0.1,..., 1.0, and then averaged.

[4] The harmonic mean of precision and recall.

ment, so that we shall use instead F_1 at one (F_1@1) and also F_1 at three and five (F_1@3, F_1@5) document level: the F_1 value obtained by assuming that the system assigns to each document either the first or the first three or five most probable classes. Both breakeven and F_1 values will be computed in micro-average (micr.) and macro-average (macr.). In all the measures, a higher value means a better performance of the model.

For comparison purposes, we have executed experiments using Naïve Bayes and the well-known Rocchio method [59], used as a perspective. In front of them, we have tested our OR gate classifiers without pruning (ML and TI versions), and both of them with three different pruning values $\alpha \in \{0.9, 0.99, 0.999\}$, noted as OR ML-$\alpha$ and OR TI-α, respectively. In all the cases, the OR classifiers used the "max heuristic" (as explained in section 3.5.3). Tables 3.1, 3.2 and 3.3 display the values of the performance measures obtained.

Several conclusions can be drawn from these experiments: the proposed OR gate models are quite competitive, frequently outperforming Rocchio and Naïve Bayes even without pruning. Particularly, the ML model seems to perform well in terms of macro averages: it gives a more balanced treatment to all the classes, and this is especially evident in those problems where the class distribution is quite unbalanced, as Reuters and Ohsumed. At the same time, the OR gate TI model performs generally well also in terms of micro averages.

The pruning procedure clearly improves the performance of both models, especially when the parameter α is high (0.99 or 0.999). The TI model with pruning seems to be, in general, the best performing model with significant differences from the other models.

3.8 Concluding Remarks and Future Works

We have described a new approach for document classification, the so called "OR Gate classifier", with different variants based on several parameter estimation methods. It is based on a Bayesian network representation which is, in some sense, the opposite that the one associated to the Naïve Bayes classifier. The complexity of the training and classification steps for the proposed model is equivalent to that of the Naïve Bayes too. In fact we can think of the OR gate model as a kind

	micr.BEP	macr.BEP	Av-11
Reuters			
NBayes	0.73485	0.26407	0.84501
Rocchio	0.47183	0.42185	0.84501
OR ML	0.66649	0.55917	0.81736
OR ML-0.9	0.67682	0.57018	0.82921
OR ML-0.99	0.69579	0.58157	0.84982
OR ML-0.999	0.70272	0.60120	0.85726
OR TI	0.76555	0.54370	0.89725
OR TI-0.9	0.75801	0.63098	0.89123
OR TI-0.99	0.80288	0.61690	0.92160
OR TI-0.999	**0.81197**	**0.63634**	**0.92445**
Ohsumed			
NBayes	0.58643	0.49830	0.76601
Rocchio	0.42315	0.44791	0.68194
OR ML	0.48017	0.58792	0.64739
OR ML-0.9	0.51504	0.55192	0.71334
OR ML-0.99	0.53939	0.56857	0.73821
OR ML-0.999	0.55494	0.57980	0.75179
OR TI	0.53122	0.56450	0.72925
OR TI-0.9	0.54300	0.57188	0.73229
OR TI-0.99	0.59350	0.59208	0.77908
OR TI-0.999	**0.60426**	**0.59730**	**0.78433**
20Newsgroups			
NBayes	0.71778	0.73629	**0.88834**
Rocchio	0.60940	0.63875	0.86583
OR ML	**0.80732**	0.81333	0.87889
OR ML-0.9	0.60460	0.63679	0.84806
OR ML-0.99	0.68346	0.69127	0.86603
OR ML-0.999	0.71844	0.73520	0.86478
OR TI	0.77689	0.79779	0.86208
OR TI-0.9	0.77126	0.77870	0.85807
OR TI-0.99	0.78221	0.78740	0.86269
OR TI-0.999	0.80165	**0.81396**	0.87832

Table 3.1: Micro and macro averaged breakeven points and average 11-point precision.

	micr.F1@1	micr.F1@3	micr.F1@5
Reuters			
NBayes	0.75931	0.49195	0.35170
Rocchio	0.70047	0.47399	0.34979
OR ML	0.67297	0.45352	0.33493
OR ML-0.9	0.67810	0.46043	0.34386
OR ML-0.99	0.70413	0.47403	0.35002
OR ML-0.999	0.71359	0.47809	0.35097
OR TI	0.75369	0.51461	0.36825
OR TI-0.9	0.74198	0.51840	0.37051
OR TI-0.99	0.78900	0.52418	0.37337
OR TI-0.999	**0.78988**	**0.52965**	**0.37550**
Ohsumed			
NBayes	0.53553	0.53979	0.42718
Rocchio	0.46064	0.46313	0.40912
OR ML	0.41676	0.46329	0.40292
OR ML-0.9	0.48459	0.49678	0.41725
OR ML-0.99	0.50294	0.51237	0.43134
OR ML-0.999	0.51768	0.51977	0.43177
OR TI	0.49048	0.50883	0.42773
OR TI-0.9	0.48528	0.51738	0.43381
OR TI-0.99	0.53700	0.55280	0.44622
OR TI-0.999	**0.53848**	**0.56440**	**0.44779**
20Newsgroups			
NBayes	0.80881	**0.48705**	**0.33212**
Rocchio	0.77233	0.48323	0.33153
OR ML	**0.81000**	0.47266	0.32614
OR ML-0.9	0.76718	0.46066	0.32072
OR ML-0.99	0.79046	0.46960	0.32459
OR ML-0.999	0.78830	0.46915	0.32461
OR TI	0.77858	0.47284	0.32695
OR TI-0.9	0.77302	0.47140	0.32644
OR TI-0.99	0.78387	0.46990	0.32511
OR TI-0.999	0.80402	0.47758	0.32835

Table 3.2: Micro F_1 values.

	macr.F1@1	macr.F1@3	macr.F1@5
Reuters			
NBayes	0.39148	0.28935	0.23116
Rocchio	0.39148	0.28935	0.23116
OR ML	0.11263	0.20092	0.26404
OR ML-0.9	0.12253	0.22908	0.27741
OR ML-0.99	0.14301	0.25440	0.28718
OR ML-0.999	0.14888	0.27181	**0.31298**
OR TI	0.45762	**0.39169**	0.30959
OR TI-0.9	0.44075	0.38363	0.29449
OR TI-0.99	0.45746	0.37399	0.27648
OR TI-0.999	**0.48456**	0.35303	0.25872
Ohsumed			
NBayes	0.40627	0.47260	0.40732
Rocchio	0.45421	0.50604	0.44945
OR ML	0.19980	0.42017	0.45870
OR ML-0.9	0.34397	0.43999	0.49561
OR ML-0.99	0.38470	0.53225	**0.50289**
OR ML-0.999	0.41318	0.54539	0.50024
OR TI	0.43602	0.53615	0.50103
OR TI-0.9	0.45655	0.55249	0.49105
OR TI-0.99	0.50015	0.56898	0.47225
OR TI-0.999	**0.50792**	**0.57549**	0.46814
20Newsgroups			
NBayes	**0.80985**	0.54983	0.39048
Rocchio	0.77095	0.54086	0.39113
OR ML	0.80880	0.58083	0.44502
OR ML-0.9	0.77119	0.55058	0.41603
OR ML-0.99	0.78987	0.54678	0.40899
OR ML-0.999	0.78589	0.53540	0.39617
OR TI	0.78682	**0.59099**	**0.44722**
OR TI-0.9	0.78163	0.59288	0.44592
OR TI-0.99	0.78558	0.57701	0.44012
OR TI-0.999	0.80603	0.56266	0.41578

Table 3.3: Macro F_1 values.

of discriminative version of Naïve Bayes, which is not a discriminative but a generative method.

According to the results of the experimental comparison carried out using several standard text collections, we found that the models without pruning can compete with Naïve Bayes, especially in terms of macro averages and in those cases where the class distribution is unbalanced. The models with pruning are even better than Naïve Bayes, getting much better results than that.

As future works, we would like to test another estimations, apart from the ML and TI approaches. For example, a Bayesian estimation could be performed, if we could combine a prior knowledge of the term and the category in the distribution. Besides, those estimations are based on a particular event model, but other alternatives could be proposed. Finally, we remark the need of a better normalization procedure, having more theoretical support (i.e. being less heuristical). We think that the models could be improved with any of these proposals.

Chapter 4

Automatic Indexing From a Thesaurus Using Bayesian Networks

4.1 Introduction

A *thesaurus* is a tool which is intrinsically designed to avoid ambiguities of any class in a document. It is composed of a set of terms with *orthogonal* meanings, along with a set of hierarchical relationships among them. A thesaurus can be very useful in different areas of Text Mining [6] and IR, removing the ambiguity, and identifying the context of a document.

Assigning some descriptors from a thesaurus to a document is a very common task for keeping organized a collection of information. Two real world examples will motivate this task, and will show the need of computerized aided methods to make these activities in a (semi)-automatic way.

1. In the parliaments of the countries of the EU (European, national and regional), all the documentation is indexed with several descriptors of the Eurovoc thesaurus (more than 6000 categories) in order to be easily accesible and searchable. This task has a very high economic cost, because it is usually done manually by teams of documentalists.

2. All the scientific medical journals indexed by MEDLINE require the articles to be indexed with their corresponding MeSH headings. This is usually a task done by the author themselves, using tools to navigate MeSH thesaurus. This is a tedious task and this method does not guarantee, for example, that the paper is indexed with the more specific headings.

For the first case, there should be available a large amount of manually indexed documents, where the descriptors assigned are obtained from the pool of descriptors selected by a team of human indexers (for example by intersecting them). In many of the cases, there will be very similar documents (parliamentary initiatives about certain laws, or certain geographical locations involved in current events), which will be finally indexed by the same descriptors. It should be desirable a system which could use all the previous learnt information and manually suggest those descriptors to the indexers. On the other hand, if no similar indexed text is found on the records, a system should also suggest descriptors based on the content, and update its knowledge with the final decisions made by indexers.

For the second case, a system, which automatically assigns MeSH headings using only information of the thesaurus (coupling terms of the paper and those from the thesaurus), could be desirable, trying to make a compromise between assigning few general MeSH headings and many specific ones.

The scope of our research is therefore, automatic subject indexing from a controlled vocabulary [51; 89] represented by the set of descriptors, and acting as a hierarchical text classifier [96; 119]. On the first introductory part, we will present in section 4.2.1 some basic definitions of thesauri (along with a example in 4.2.2), a formal characterization in section 4.2.3, a list of real-world thesauri in 4.2.3, and the standard formal language to manually define them.

After the introduction, we will introduce the task of Thesaurus Based Automatic indexing, where we shall clearly state this problem and their difficulties (sections 4.3.1 and 4.3.2). We will review some approaches in the literature to similar problems, present a baseline and discuss the possible evaluation measures (sections 4.3.4 to 4.3.5).

The sections concerning the Bayesian network are organized as follows: after a brief introduction (given in section 4.4) we describe the proposed Bayesian network model of a thesaurus in section 4.5, whereas the extension of the model to cope with training data is presented in section 4.6. The experimental evaluation is explained in section 4.7. Finally, section 4.8 ends this chapter containing the final remarks and some proposals for future work.

4.2 Basics of Thesauri

4.2.1 Definitions

Broadly speaking, a *thesaurus* consists of a set of terms (which can be relevant to a certain domain of knowledge) and a set of relationships between them. Its main aim is to represent concepts without ambiguity in order to avoid confusion and misunderstanding.

The basic unit of a thesaurus is often called the *descriptor*[1]. A descriptor is a word or phrase which identifies an important notion in a certain domain of knowledge, i.e. it designates essential entities in the area covered by the thesaurus.

A thesaurus is not always tied to a closed area of knowledge. In fact, following [127], we can divide thesauri into two different kinds: *conceptual thesauri*, with abstract and conceptual terms as descriptors (e.g. Eurovoc), and *natural language thesauri*, which tend to be more specific covering a certain area of knowledge (e.g. AGROVOC or MeSH).

Other basic thesaurus units are the *non-descriptors*. These are words or expressions which basically denote the same notion as a descriptor in the thesaurus language.

Besides, a thesaurus also includes the following three types of semantic relationships:

- *Hierarchical relationships*, involving one descriptor and a more specific or broader one.

[1]It is also sometimes called a *concept* or an *index term*.

- *Equivalence relationships*, between non-descriptors and descriptors, listing the equivalent terms for a certain concept, and the possible uses of a descriptor. The equivalence relationships may in fact cover relationships of various types: identical, similar or opposite meanings, and even inclusion.
- *Associative relationships*, between descriptors, which can also be of various kinds: cause and effect, agency or instrument, concomitance, constituent elements, location, etc. They generally link two descriptors that do not meet the criteria for either equivalence or hierarchical relationships, and are used to suggest another descriptor that would be helpful for the thesaurus user to search with.

In the first case, the hierarchy defined by the thesaurus specifies the BT (*broader term*) and NT (*narrower term*) relationships. The NT designates a more specialized descriptor for a particular one. For each BT, there is the corresponding NT relationship (they are reciprocal). In other words, if the broader term of "A" is "B", then the narrower term of "B" is "A". If a descriptor has no broader term, it is sometimes called a *top term*. On the other hand, if a descriptor has no narrower term, it is often called a *basic descriptor*.

In the second case, the equivalence relationships are UF (*used for*) and USE: UF between the descriptor and the non-descriptor(s) it represents, and USE between a non-descriptor and the descriptor which replaces it.

For the third case, a thesaurus specifies a *related term* relationship (RT). This is a symmetrical relationship: if a descriptor "A" is related to "B" by means of an RT relation, then "B" and "A" should also be in RT.

Certain thesauri also include a *scope note* (SN) for several descriptors which defines or limits the specific use of a term, although they are usually addressed to manual indexers.

4.2.2 A Small Example

We give here an example, based on a real thesaurus (Eurovoc), which will be reviewed later. Eurovoc is a multilingual thesaurus that provides a means of indexing the documents in the documentation systems of the European institutions and of their users. We show here a fragment of it.

Figure 4.1 displays the BT relationships between some descriptors of Eurovoc and the USE relationships between the non-descriptors and these descriptors. In the example there are two complex descriptors, *health service* and *health policy*, and three basic descriptors, *medical centre*, *medical institution* and *psychiatric institution*.

Health service is the broader term of *medical centre*, *medical institution* and *psychiatric institution*; *health policy* is in turn the broader term of *health service* and also of other five descriptors which are not displayed[1]. The associated non-descriptors are: *medical service* for *health service*; *health* and *health protection* for *health policy*; *dispensary* and *health care centre* for *medical centre*; *clinic*, *hospital* and *outpatients' clinic* for *medical institution*; and *psychiatric hospital* for *psychiatric institution*.

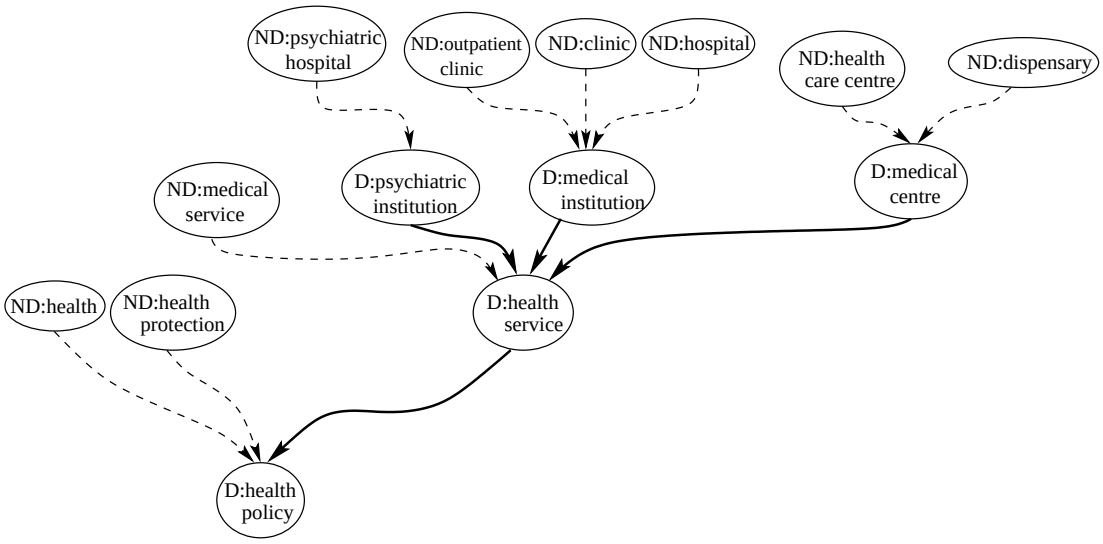

Figure 4.1: BT (bold lines) and USE (dashed lines) relationships for the descriptors and non-descriptors in the example about *health*.

[1]These non displayed descriptors are *health care system*, *health costs*, *health expenditure*, *health statistics* and *organisation of health care*.

4.2.3 A Formalization of Thesauri

Here, we give a mathematical formulation of the main existing relationships in a thesaurus. This is only the formalization of the previous explanation and, in posterior developments, we shall assume that any thesaurus follows this model.

We firstly state that a thesaurus may be formalized as an eight-tuple:

$$(\Omega, \Delta, \Gamma, W_\Delta, W_\Gamma, \mathrm{USE}, \mathrm{RT}, \mathrm{BT}),$$

where the sets $\Omega = \{\omega_1, \ldots, \omega_n\}$, $\Delta = \{\delta_1, \ldots, \delta_m\}$ and $\Gamma = \{\gamma_1, \ldots, \gamma_k\}$ represent the terms (words), descriptors and non-descriptors in the thesaurus, respectively.

There is a map $W_\Delta : \Delta \to 2^\Omega \setminus \{\emptyset\}$, where 2^Ω is the set of all subsets of Ω. Similarly, there is another map $W_\Gamma : \Gamma \to 2^\Omega \setminus \{\emptyset\}$. Clearly, $W_\Delta(\delta)$ (resp. $W_\Gamma(\gamma)$) denotes the set of terms associated to a descriptor δ (respectively a non-descriptor γ).

There is also a map $\mathrm{USE} : \Gamma \to \Delta$, such that $\forall \gamma \in \Gamma$, $\mathrm{USE}(\gamma) \in \Delta$ is the descriptor associated with the non-descriptor γ. Therefore, the inverse map UF can also be defined as $\mathrm{UF}(\delta) = \mathrm{USE}^{-1}(\delta)$ (the set of non-descriptors associated with the descriptor δ).

The relation map $\mathrm{RT} : \Delta \to \Delta$ is defined between pairs of descriptors, verifying $\forall \delta_1, \delta_2 \in \Delta$, if $\mathrm{RT}(\delta_1) = \delta_2 \Rightarrow \mathrm{RT}(\delta_2) = \delta_1$. Thus, the binary relation corresponding to RT is symmetrical.

Finally, there is another function $\mathrm{BT} : \Delta \to \Delta \cup \{\emptyset\}$ (where $\emptyset$ represents the empty descriptor) such that $\mathrm{BT}(\delta) \in \Delta$ is the broader descriptor containing δ and $\mathrm{BT}(\delta) = \emptyset$ means that the descriptor δ is not contained in a more general one (i.e. it is a *top term*). $\mathrm{NT}(\delta) = \mathrm{BT}^{-1}(\delta)$ is the set of narrower (more specific) descriptors which are contained in δ.

If $\mathrm{BT}^{-1}(\delta) = \emptyset$, then the descriptor δ does not contain a more specific descriptor (it is a *basic descriptor*). If $\mathrm{BT}^{-1}(\delta) \neq \emptyset$, we say that δ is a *non-basic* or *complex descriptor*. More generally, when a descriptor δ is *polyhierarchical* (as in Eurovoc, for instance), then $\mathrm{BT}(\delta)$ is not a single descriptor but a subset of the set of descriptors and hence, BT is not a function but a correspondence.

Note that the function/correspondence BT must satisfy the following property:

```
1191 CALATHEA
1192 CALATHEA ALLOUIA
1193 CALATHEA LUTEA
1194 CALCAREOUS SOILS
1195 CALCITONIN
1196 CALCIUM
1197 CALCIUM CARBONATE
1198 CALCIUM CYANAMIDE
1199 CALCIUM FERTILIZERS
1200 CALCIUM HYDROXIDE
1201 CALCIUM NITRATE
1202 CALCIUM OXIDE
1203 CALCIUM SULPHATE
```

Figure 4.2: Fragment of the list of descriptors in the AGROVOC thesaurus.

$$\mathrm{BT}\overbrace{\circ\cdots\circ}^{k}\mathrm{BT}(\delta) \neq \delta,\ \forall\delta \in \Delta, \forall k = 1, 2, \ldots,$$

where $\circ$ represents composition. This property guarantees that the BT relationships constitute a true hierarchy, avoiding, for example, the existence of cycles in it.

4.2.4 Defining Thesauri with a Standard Language

Most of the thesauri mentioned on this work are public and they are freely available for non-commercial purposes. Thus, they can be obtained in several non-standard formats: XML, ASCII (plain text), PDF, etc. Defining a thesaurus following some standard language is very important if we want to build a generic software application which does not rely on specific thesauri. This application can be aimed to, for instance, navigating and consulting the thesaurus, easily obtain descriptors and relationships, or aiding in the indexation of texts with the descriptors of the thesaurus (which we shall discuss later).

Currently there is an accepted standard: the Simple Knowledge Organisation System or SKOS [90], developed by the W3C, is a family of formal languages for thesaurus representation, structured controlled vocabulary and any other taxo-

nomy. SKOS is built on RDF and RDFS, and its main objective is to enable easy publication of controlled structured vocabularies for the Semantic Web. SKOS is currently under a proposed recommendation for the W3C since the first quarter of 2008.

4.2.5 Real World Thesauri

There are several industrial and sociopolitical thesauri, used in different domains of knowledge. Here, some of them are described, along with their special characteristics. The table 4.1 gives an idea of the size (number of descriptors, relationships and non-descriptors) of some of these thesauri.

4.2.5.1 Eurovoc

Eurovoc [48] is an official thesaurus of the European Union. It covers the fields in which the European Comunities are active, being its main aim to provide a means of indexing the official documents in the documentation systems in the institutions. It is used by the European Parliament, the Office for Official Publications of the European Communities, the national and regional parliaments in Europe, and other non-EU countries.

Eurovoc is a plurilingual thesaurus, which means there exist different editions of it, in the 21 official languages of the Union. All of them share the same descriptors, making it easy to provide "conceptual indexation" and search of a document, no matter the language it is written in.

The first level of the hierarchy is called the tematic field, being 21 in Eurovoc. The second level is the microthesaurus (it contains 127 microthesauri), and each tematic field is composed of certain microthesauri.

In its version 4.2, it comprises 6645 descriptors, with 512 top terms. For the relationships, it contains 6669 of the hierarchical kind (BT/NT), and 3636 RT.

Some of the descriptors are polyhierarchical (they can have more than one descriptor as a broader term), and RT relationships are, by definition, incompatible with hierarchical ones.

It is a conceptual thesaurus, in the sense that very different fields in the sociopolitical life (geographical areas, medical terminology, agriculture,...) can be indexed with it.

4.2.5.2 AGROVOC

AGROVOC [4] is a multilingual, structured and controlled vocabulary designed to cover the terminology of all subject fields in agriculture, forestry, fisheries, food and related domains (e.g. environment). It is been developed since 1980 by the Food and Agriculture Organization of the United Nations (FAO), and has been translated into 17 languages. It is free for non-commercial use.

Its structure is similar to Eurovoc, using a new relationship called "spatially included in" (used only for geographical descriptors).

In its English edition, AGROVOC contains 28435 descriptors and 10932 non-descriptors. 32176 BT/NT and 27589 RT relationships can be found on it. A small fragment of this thesaurus can be seen in figure 4.2.

AGROVOC is used all over the world (about ninety countries), mostly for indexing and retrieving data in agricultural information systems.

4.2.5.3 NAL Thesaurus

The NAL Agricultural Thesaurus [98] (NALT) was built in 2002 by the National Agricultural Library to meet the needs of the USDA, Agricultural Research Service (ARS). The subject scope of agriculture is broadly defined in the NALT, and includes terminology in the supporting biological, physical and social sciences. Biological nomenclature comprises a majority of the terms in the thesaurus and is located in the "Taxonomic Classification of Organisms" Subject Category. Political geography is mainly described at the country level.

This thesaurus is currently used at several sites. In 2003, NAL implemented the thesaurus as the controlled indexing vocabulary for their in-house bibliographic database, AGRICOLA, in conjunction with the implementation of a new electronic library management system. In 2004, E-Extension, the prototype database of state cooperative extension publications, chose the NAL Agricultural Thesaurus as their controlled vocabulary. Other non-USDA sites use

the thesaurus, such as the Agricultural Network Information Center (AgNIC), which adopted it in 2002. AgNIC partners use the thesaurus as a vocabulary for indexing and as a search aid to their web sites through web services on their portal.

It is organized into 17 subject categories, indicated by the "Subject Category" designation in the thesaurus. This organization can be used to browse the thesaurus in a specific discipline or subject area. It includes 42326 descriptors, 25985 non-descriptors, 44545 BT/NT hierarchical relationships, and 17324 symmetrical RT relationships.

The NAL Agricultural Thesaurus is also freely available online from NAL website and as a web service to other web-connected programs.

Name	**Eurovoc**	**AGROVOC**	**NAL**
Descriptors	6645	28435	42326
Relationships (BT/NT+RT)	10305	59765	61869
Non-descriptors	6769	10932	25985

Table 4.1: Comparative numbers for real thesauri: the number of descriptors and non-descriptors (only English ones, if the thesaurus is multilingual) is shown, together with the number of relationships (the size of the graph the thesaurus represents).

4.2.5.4 MeSH

MeSH [101] is a significant thesaurus in biomedicine, built by the American National Library of Medicine (NLM) in 1960. It has been updated subsequently and translated into many languages. It has three main aims: indexing biomedical literature (published in Index Medicus), cataloging monographs and multimedia content stored in the NLM and presenting a vocabulary for the Index Medicus user, in order to make precise and accurate searching over the index.

MeSH contains over 22000 *headings* (descriptors in MeSH terminology) and the three types of relationships described before: hierarchical or BT/NT (MeSH is divided into 15 *trees*), synonymous (non-descriptors are called *entry terms* here), and related terms.

MeSH has a tree structure, which determines the level of superiority or inferiority to individual descriptors and establishes hierarchical relationships between conceptually related terms. Although the basic concepts explained before are present on MeSH, it includes several aditional features that makes MeSH different from other thesauri:

- *Check tags:* tags to describe the attribute of the study.
- *Publication type:* type of publication or type of study.
- *Related concepts:* several added semantic relationships.

An introduction to the MeSH thesaurus and, in general, biomedical information can be found in [54].

4.3 Thesaurus Based Automatic Indexing

4.3.1 Statement of the Problem

We define the problem of automatic indexing in a thesaurus-based set of categories as a problem of text categorization on the set of classes defined by a thesaurus (the set of descriptors), using or not the information contained on the thesaurus (terms and relationships).

Thus, we can propose three different approaches to the problem:

1. A solution which only uses *information from the thesaurus*.
2. A solution which uses *information from preclassified documents*.
3. A solution which uses *both* kinds of information.

A system implementing the first solution (a *document clustering* problem, in the sense of Golub [51]) should be capable, given a thesaurus and a document, to suggest descriptors from the thesaurus to index the document. For the second (a classical problem of supervised Text Categorization using Machine Learning methods) and third solution, we need to have a previously indexed corpus of

documents. The third approach, which is a *mixed approach* between the first and the second one, should combine the expert and previous knowledge given by the preclassified corpus, and the static knowledge given by the taxonomy of the thesaurus and its relationships.

The second approach probably is not the best option. We could have treated this problem as a simple Machine Learning problem. However, the corpus availability is not sometimes very easy, and presents the following problems:

- The preclassified corpus should contain *at least one document indexed for each descriptor* (which will make the corpus a large one, because the number of categories of a thesaurus is much larger than the number of categories of the state-of-the-art Text Categorization corpora). In order to make good estimations, the number of documents assigned to each category should also be reasonable (it is not enough with one or two documents). This is not regrettably possible because sometimes we have a corpus available which does not cover at all the whole set of categories of the thesaurus.

- Ignoring hierarchical relationships (and the RT ones) makes all the categories independent, which is not obviously true for this problem. From a certain point of view, a document indexed with a descriptor belongs to all the set of more general descriptors.

- For this problem, there are several keywords which may generate some automatic descriptor assignments, and they could not be inferred from data (for example, a document refering to "Madrid" could be automatically indexed by the descriptor "Spain" if that relationship appears in the thesaurus).

Because thesauri are mainly used to add several "index terms" to documents the problem must be, in all of the cases, a multilabel categorization approach. However, it is a clear fact that this is a very difficult problem from a "multilabel" point of view (the results are expected to be rather poor), and a category-ranking approach could be better for a semi-automated approach.

4.3.2 Difficulties of the Problem

This peculiar problem differs from a "classical approach" of Text Categorization presenting the following difficulties:

- **The high dimensionality of the set $\mathcal{C}$ of categories:** as we could see in the description of the real world thesauri, presented on the previous section, we are managing several thousands categories. The existing test collections for Text Categorization (where the existing approaches are tested) have a few number of categories (none of the classical test collections are over one thousand categories, being most of them under one hundred).

- **The lack of data test collections:** up to date, there is not a freely available test corpus of documents classified over a thesaurus (except perhaps, OHSUMED [55]). This makes very difficult the task of testing new approaches, because of the absence of a "standard" comparison procedure.

- **The problem of feature selection:** a thesaurus itself contains a lot of information. Which is useful to make more accurate classification? Should certain kinds of relationships not be considered? Should non-descriptors terms be considered as a source of meta-information for a certain category? Apart from that, standard feature selection methods for documents could not be useful here, due to the hierarchical relationships in the set of classes (two classes can be disjoint, in the sense that they are different, but semantically related by an "is a" or other semantic relationship).

- **The problem of comparison:** when discussing a new categorization method, a *baseline* is a very simple approach for the problem it is being solved, which is supposed to be worse than our proposal. Sometimes classical methods as Naïve Bayes [83] are used for this purpose. Other times, a well-working model is used as this baseline. Which is the suitable baseline for this kind of experimentation? In 4.3.4 we present a very simple solution for the first approach to the problem stated on 4.3.1 (a solution which only makes use of thesaurus information).

- **The problem of evaluation:** this is a very important point, to make conclusive results. If the problem is considered to be a category-ranking problem, a very common way to evaluate are the precision/recall measures, and the MAP measure (the average precision on the standard eleven points of recall). We shall discuss later, in section 4.3.5, the advantages and drawbacks of every evaluation measure, selecting the more appropriate to evaluate this problem.

  Obviously, the consideration of this evaluation procedure is not clearly satisfying the semantic links among categories. For example, imagine that a document into the test set is classified to a category A, which is the broader term of B. If A does not appear in the results of the correct classes of this document, but B does, using pure precision/recall evaluation, the category A will be considered as wrong, even when it is almost the desired category, with a certain degree of generality.

- **The problem of related class influence:** as in hierarchical classification systems, a class can be associated to a certain document by means only of hierarchy-related classes. For example, if all the narrower terms of a descriptor seem to be relevant for a certain document, the system should decide returning the broader descriptor instead. This fact means that the training set does not need to be complete (in the sense of containing at least one document of each class), and so, a certain descriptor can be returned in the results, even if it does not appear in the entire training set.

4.3.3 Related Work in Automated Indexing

We give here some references for related works on automatic indexing, where the set of categories belongs to a thesaurus:

- In [93], an indexer is built for the DESY [40] thesaurus (a thesaurus in the high energy physics domain used at the CERN) using several classification methods. This work is also presented, in a shorter form, in [92].

- An old approach to indexing in Eurovoc, without considering the structure of the thesaurus can be found in [70].

- In [125], the automatic indexer for Eurovoc used in the European Parliament is described and evaluated. In [124] and [127], the same authors propose an algorithm for automatic indexing of Eurovoc documents in a multilingual environment. This method of automatic indexing is also used as a tool to calculate multilingual document similarity in [126].

- In [87] an algorithm to assign keywords from a thesaurus is presented and tested in AGROVOC. It is also shown in [89] and in [88].

4.3.4 A Simple Baseline: a Modified Vector Space Model

In this section we present a baseline for the problem of unsupervised classification (i.e. given a document and a thesaurus, associate to the document the most probable descriptors). Due to the lack of a training set, this procedure needs to be done comparing the text of the descriptors (and not descriptors) to the text of the document. Roughly speaking, a method of this kind will return with higher confidence values those descriptors which have less distance to the document.

The easiest way to measure the distance between two documents is the Vector Space Model (VSM) which is one of the most successful models in IR. It was firstly proposed by G. Salton, A. Wong and C. S. Yang [115] in the 70s, and it is used until nowadays. Each document is represented as an n-tuple of nonnegative real numbers. Moreover, each coordinate corresponds to one of the n different terms present in the collection. Thus, the value of the coordinate describes the importance of the term in the document, i. e., a very important term in the document has a higher value of the coordinate than less important ones. The value (weight) of the importance of the term i on the document j (the i-th component of the j-th document vector) is denoted by w_{ij}, and it is often defined as:

$$w_{ij} = \mathrm{tf}_{ij}\ \mathrm{idf}_i = \frac{f_{ij}}{\max_k f_{kj}}\ \log\frac{D}{D_i},$$

where f_{ij} is the absolute frequency of the i-th term in the j-th document. On the other hand, idf_i stands for inverse document frequency of i-th term, as the

logarithm of the number of documents between the number of documents that term appears in, and it measures the rarity of the term in the collection.

Note that this scheme for the values w_{ij} is not unique, and G. Salton and C. Buckley proposed several variants in [116]. Anyway, the setting used here is a standard one.

Finally, similarity between two documents is defined as the cosine of the angle between the two vectors:

$$\text{sim}(d_i, d_j) = \frac{\langle d_i, d_j \rangle}{\|d_i\| \|d_j\|} = \frac{\sum_{k=1}^{n} w_{ki}\, w_{kj}}{\sqrt{\sum_{k=1}^{n} w_{ki}^2 \sum_{k=1}^{n} w_{kj}^2}}$$

Applying the VSM to the case of unsupervised classification over the descriptors of a thesaurus is very simple and can be done using the following algorithm:

1. Each descriptor is associated to a vector in the space (it is represented the same way as a document).

2. When classifying a new document d, descriptors are ranked in decreasing order of similarity $\text{sim}(d, \text{desc}_i)$, for all descriptors desc_i of the thesaurus.

Building the vectors for the descriptors (the first step of the algorithm) can be done in the two following ways:

- **VSM with independent classes:** [73; 135] we associate to each descriptor the terms of its own descriptors, and the ones belonging to its non-descriptors (without any kind of distinction). Obviously, no relationships between classes are taken into account.

- **Hierarchical VSM: (HVSM)** [2; 3] each descriptor is represented by the vector containing its own terms, the terms of its associated non-descriptors, and the terms of the corresponding broader term added recursively, until a top term is found[1]. Hierarchical VSM is supposed to be more accurate, because BT/NT relationships are used.

[1] We can describe this model thinking that every descriptor "contains" its own information, and all the information contained by its broader term, recursively. So, a descriptor is the "specialization" of its broader term.

This approach is very similar to the one used in [3], for a problem of hierarchical classification of documents, adapted to the case of the set of categories defined by a thesaurus.

4.3.5 Evaluation of the Task

Here we explore the different alternatives to evaluate this task, commenting the advantages and drawbacks of each one. Based on these conclusions, we shall use some chosen measures for the evaluation of the experiments which will be carried out in the empirical test of the models.

To begin with, we have stated that this is a multilabel document categorization problem. We can suppose that, without loss of generality, we can make traditional multilabel evaluation. The main two measures used to evaluate this task on the literature [119] are the precision/recall breakeven point and the F_α with $\alpha = 1$, both in their micro and macro versions. The first one (breakeven point) can be used with a classifier which returns a CSV[1] value for each pair (document, category), and the second one (F_1 per category) needs a "hard" classification (an assignment of a subset of the set of categories, $\mathcal{C}$, to each document). Because we shall be using probabilistic classifiers, we shall not do hard classification and then, F_1 per category could not be used.

The two previous measures, F_1 and precision/recall breakeven point are used from the viewpoint of the categories, i.e. they are the average of the individual F_1 and precision/recall breakeven point obtained for each one of the categories, considered as a binary problem. Knowing that the set of documents in such a corpus tends to be very unbalanced, and maybe many categories do not get any assignments, the classical methodology, is not appropriate for the evaluation of the real task performed by the human indexer.

But, what is the more natural approach to the task of indexation on this kind of environment? Consider a trained model, and a certain document d to categorize. It should be desirable to present a list of descriptors to the indexer

[1] We recall that the CSV is a real number that measures the confidence of how a document is assigned to a category by the classifier (the greater this number is, the bigger confidence the classifier reports).

in order to help him make his decision. This can be done in two different (and independent) ways:

1. We present the indexer the whole list of descriptors, ranked by their probability value $p(c_i|d)$, obtained with the model. If the model is good, the indexer would not need to traverse the whole list to find all the interesting list of descriptors, because they will be on the top of the list. Alternatively, there should be few "noise" (incorrect assignments) on this top part of the list.

2. Another alternative is to take the previous list, and cut it at a certain position, assuming that, although we could lose some interesting descriptors, the returned set would be accurate.

To evaluate the first proposal, we can use an IR approach. We think a document to be categorized as a query given to an IR system. The list of returned descriptors will be the list of documents returned by the IR system, and the list of true assignments would be the relevance judgements corresponding to that query. Thus, we can compute, for each document, a precision/recall curve, in order to obtain, finally, an averaged precision/recall curve for the set of documents to categorize. This curve can be summed up with the well-known mean average precision (MAP) on the standard 11 points of recall: $(0.0, 0.1, 0.2, \ldots, 0.9, 1.0)$.

The second proposal can be evaluated in a similar way. In this case, we do not have a list of ranked descriptors, but a set of proposed assignments (cutting the previous list at a certain given position). The resulting set will have a precision and a recall value, with respect to the true assignments to the document, and both values can be combined with the F_1 measure. Finally, all the F_1 values can be averaged over the set of documents. Note that this F_1 "per-document" is not the same value than the "per-category" one. Besides, the choice of the position of the cut in the list will affect this value. Consequently, we shall use $F1@X$ to denote the F_1 value obtained when the list of ranked descriptors is cutted in the Xth descriptor, taking the first X as proposed assignments, and discarding the rest. This measure can also be, obviously, "micro" or "macro" averaged.

Note that any of the three proposed alternatives (precision/recall breakeven point, MAP and $F1@X$) give values between 0 and 1, and represente a better performance of a model, when they are higher.

4.4 A Bayesian Network Model for Automatic Thesaurus Indexing

We present here two different categorization models to solve the problem of indexing a set of documents over the set of descriptors belonging to a thesaurus. The model presented here is basically the same we showed on [22]. The first of the models has the characteristic that no training is required to start using the system (it uses only information from the thesaurus). Thus, we shall exploit only the hierarchical and lexical information from the thesaurus to build the classifier. As we stated in section 4.3.1, this is an advantage because the model is ready-to-use with almost any thesaurus and without having preclassified documents (in a large hierarchy, the amount of preclassified documents necessary for training may be huge). On the other hand, this is also a weakness because any kind of information not considered in the thesaurus (e.g. other relations, specific information handled by documentalists,...) will not be taken into account and, therefore, we should not expect very high success rates in comparison with classifiers that are built using training data [28; 45; 74; 112].

In this sense our initial proposal is more similar to the work in [2; 3], where a method to populate an initially empty taxonomy is presented. The working framework is that a documentalist would prefer to confirm or discard a given classification hypothesis proposed by the system rather than examining all the possible alternatives.

However, the model can also naturally incorporate training data in order to improve its performance: this leads to the second model, which can be seen as a natural extension of the first. The information provided by preclassified documents can be appropriately merged with the hierarchical and equivalence relationships among the descriptors in the thesaurus, in order to obtain a classifier better than the one we would obtain by using only the training documents.

Another important characteristic of our model is that is based on Bayesian networks. To the best of our knowledge, no Bayesian network-based models other than Naïve Bayes have been proposed to deal with this kind of problems [67]. We create a Bayesian network to model the hierarchical and equivalence relationships in the thesaurus, and next we extend it to also use training data. Then, given a document to be classified, its terms are instantiated in the network and a probabilistic inference algorithm, especifically designed and particularly efficient, computes the posterior probabilities of the descriptors in the thesaurus.

4.5 The Basic Model: The Bayesian Network Representing a Thesaurus

In this section we describe the Bayesian network model proposed to represent a thesaurus, including the graphical structure, the conditional probabilities and the inference mechanism.

4.5.1 Bayesian Network Structure

A simple approach for modeling a thesaurus as a Bayesian network would be to use a type of representation directly based on the graph displayed in the figure 4.1, containing descriptor and non-descriptor nodes, then adding term nodes representing the words in the thesaurus and connecting them with the descriptor and non-descriptor nodes that contain these words. This would result in a network structure as the one displayed in the figure 4.3. The problem with this type of topology is that each descriptor node receives two or three kinds of arcs with different meaning, those from its non-descriptor nodes and those from its term nodes and, for the case of complex descriptor nodes, also those arcs from the narrower descriptor nodes that they contain.

As this would make much more difficult the process of assigning the associated conditional probability distributions to the nodes, we propose a different topology. The key ideas are:

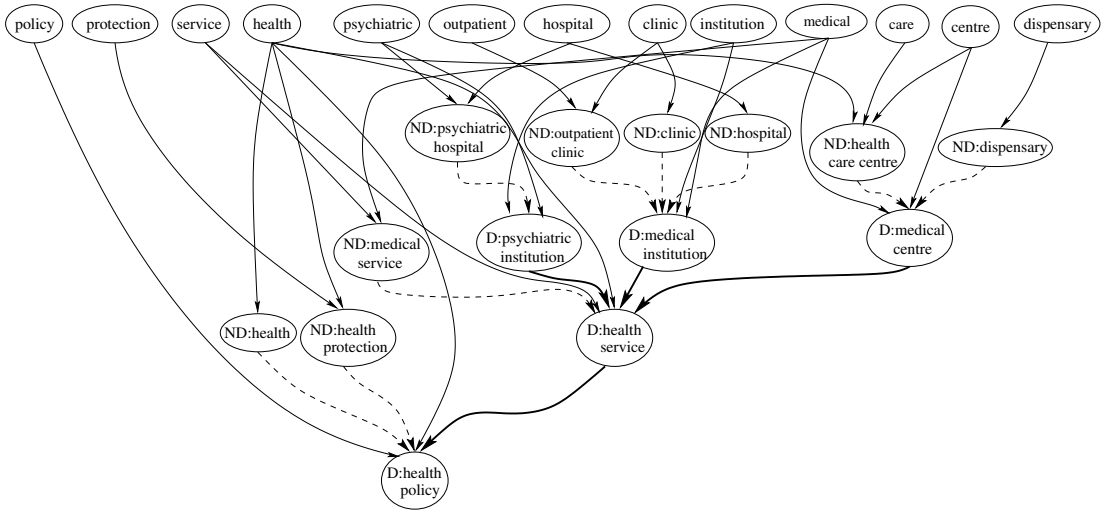

Figure 4.3: Preliminary Bayesian network in the example about *health*.

(1) to explicitly distinguish between a concept and the descriptor and non-descriptors used to represent it, and,

(2) to clearly separate, through the use of additional nodes, the different information sources (hierarchy and equivalence relationships) influencing on a concept.

According to the first key idea, each concept, labeled identically as the descriptor representing it, will be a node C in the network. We shall also distinguish between basic and complex concepts: the former do not contain other concepts, whereas the later are composed of other concepts (either basic or complex). Each descriptor and each non-descriptor in the thesaurus will also be nodes D and ND in the network. All the words or terms appearing in either a descriptor or a non-descriptor will be term nodes T. To accomplish with the second key idea, for each concept node C we shall also create two (virtual) nodes: E_C, which will receive the information provided by the equivalence relationships involving C; and H_C, which will collect the hierarchical information, i.e. the influence of the concepts contained in C.

With respect to the links, there is an arc from each term node to each descriptor and/or non-descriptor node containing it. There are also arcs from each

non-descriptor node, associated to a concept node C, to the corresponding virtual node E_C (these arcs correspond with the USE relationships), as well as from the own descriptor node representing the concept C to E_C. There is also an arc from each concept node C', excluding those nodes which are associated with a top descriptor, to the virtual node(s) H_C associated with the broader complex concept(s) C containing C' (these arcs correspond with the BT relationships). Finally, there are arcs from the virtual nodes E_C and H_C to its associated concept node C.

We shall denote $\mathcal{T}$ the set of term nodes, $\mathcal{DE}$ and $\mathcal{ND}$ the sets of descriptor and non-descriptor nodes, respectively, $\mathcal{C}$ the set of concept nodes, and $\mathcal{E}$ and $\mathcal{H}$ the sets of virtual equivalence and hierarchical nodes, respectively. All the nodes will represent binary random variables. The domain of each variable is: $\{t^+, t^-\}$ $\forall T \in \mathcal{T}$; $\{de^+, de^-\}$ $\forall DE \in \mathcal{DE}$; $\{nd^+, nd^-\}$ $\forall ND \in \mathcal{ND}$; $\{c^+, c^-\}$ $\forall C \in \mathcal{C}$; $\{e^+, e^-\}$ $\forall E \in \mathcal{E}$; $\{h^+, h^-\}$ $\forall H \in \mathcal{H}$. For term nodes, their values indicate whether the term appears in the document to be classified. For descriptor and non-descriptor nodes, the values represent whether the corresponding descriptor or non-descriptor may be associated with the document. For concept nodes and their associated virtual nodes the values mean whether the concept is appropriate/relevant to classify the document. $Pa(X)$ will represent the parent set of a node X in the graph. The proposed network topology is completely determined by specifying the parent set of each node: for each term node $T \in \mathcal{T}$, $Pa(T)$ is the empty set; for each descriptor and non-descriptor node $DE \in \mathcal{DE}$ and $ND \in \mathcal{ND}$, $Pa(DE)$ and $Pa(ND)$ are in both cases the set of term nodes associated with the words that appear in DE and ND, respectively; for each virtual equivalence node $E_C \in \mathcal{E}$, $Pa(E_C)$ is the set of descriptor and non-descriptor nodes that define the concept C; for each virtual hierarchical node $H_C \in \mathcal{H}$, $Pa(H_C)$ is the set of concept nodes contained in the corresponding complex concept C; finally, for each concept node $C \in \mathcal{C}$, $Pa(C) = \{E_C, H_C\}$, the set of its two associated virtual nodes. For the previous example the corresponding subnetwork is shown in Fig. 4.4. It should be noticed that this model is slightly different and more general than the one we proposed in [17].

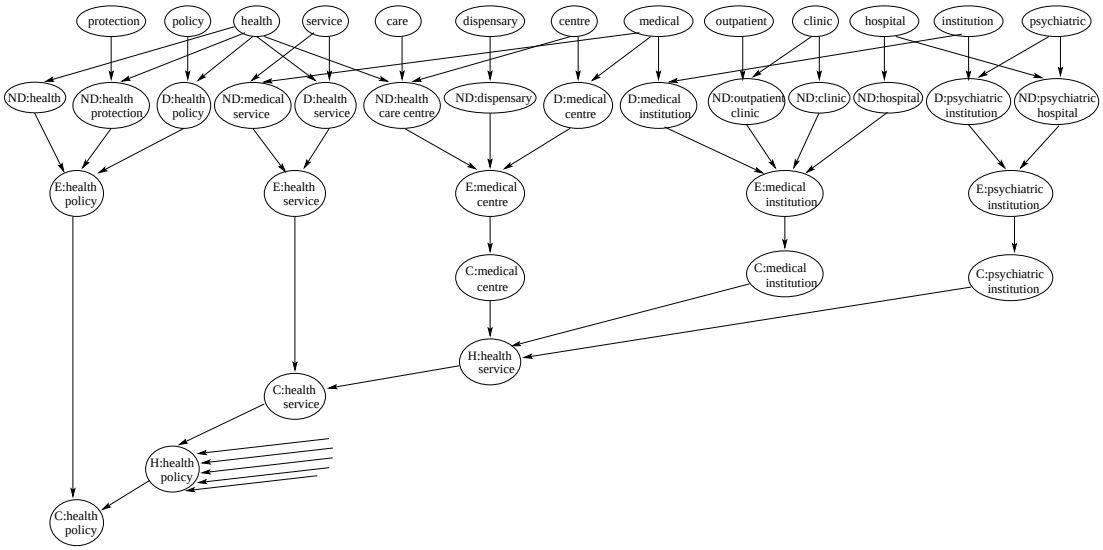

Figure 4.4: Bayesian network in the example about *health*.

4.5.2 Conditional Probability Distributions

The probability distributions that must be specified are the prior probabilities for term nodes, $p(t^+)$, and the conditional probabilities for the remaining nodes: $p(de^+|pa(DE))$, $p(nd^+|pa(ND))$, $p(c^+|pa(C))$, $p(e^+|pa(E))$ and $p(h^+|pa(H))$. In all the cases $pa(X)$ represents a configuration of the parent set $Pa(X)$ of the node X.

For the prior probabilities of term nodes we propose using a constant value, $p(t^+) = p_0$, $\forall T \in \mathcal{T}$ (although we shall see later that this is not an important issue at all).

As the treatment of the descriptor and non-descriptor nodes will the same, in order to simplify the exposition, from now on we shall denote $\mathcal{D} = \mathcal{DE} \cup \mathcal{ND}$ and we shall refer to both descriptor and non-descriptor nodes as descriptor nodes. An element in $\mathcal{D}$ will be denoted as D. For the conditional probabilities of a descriptor node D given the terms that it contains, $p(d^+|pa(D))$, we propose using a canonical additive model [15], which has been successfully employed in

the IR field:

$$\forall D \in \mathcal{D},\ p(d^+|pa(D)) = \sum_{T \in R(pa(D))} w(T, D), \tag{4.1}$$

where $w(T, D)$ is the weight associated to each term T belonging to the descriptor D. $R(pa(D))$ is the subset of parents of D which are observed in the configuration $pa(D)$, i.e., $R(pa(D)) = \{T \in Pa(D) \,|\, t^+ \in pa(D)\}$. So, the more parents of D are observed the greater its probability of relevance. These weights can be defined in any way, the only restrictions are that $w(T, D) \geq 0$ and $\sum_{T \in Pa(D)} w(T, D) \leq 1$.

To define the weight of a term in a descriptor, $w(T, D)$, we propose a normalized tf-idf scheme, as those frequently used in IR:

$$w(T, D) = \frac{tf(T, D) * idf(T)}{\sum_{T' \in Pa(D)} tf(T', D) * idf(T')}.$$

The *inverse descriptor frequency* of a term, $idf(T)$, is

$$idf(T) = \ln\left(\frac{m}{n(T)}\right),$$

where $n(T)$ is the number of descriptors and non-descriptors in the thesaurus that contain the term T and m is the total number of descriptors and non-descriptors. The *term frequency* of a term in a descriptor, $tf(T, D)$, is the number of times that this term appears in the descriptor (which will be almost always equal to 1, because the descriptors usually contain very few words).

For the conditional probabilities of each virtual equivalence node E_C given the descriptor nodes that define the concept C, $p(e_c^+|pa(E_C))$, it is not appropriate to use the previous additive model, because each descriptor alone is supposed to be able to represent the concept, and this behaviour cannot be obtained using an additive model. So, we propose to use another kind of canonical model, namely an OR gate [102]:

$$\forall E_C \in \mathcal{E},\ p(e_c^+|pa(E_C)) = 1 - \prod_{D \in R(pa(E_C))} (1 - w(D, C)). \tag{4.2}$$

$R(pa(E_C)) = \{D \in Pa(E_C) \,|\, d^+ \in pa(E_C)\}$ and $w(D, C)$ is the probability that

the descriptor D alone (the other descriptors being non relevant) makes concept C relevant, with $0 \leq w(D, C) \leq 1$.

For the weights of the descriptors in the concepts, $w(D, C)$, a reasonable choice is a high value near 1.0, because any descriptor associated with a concept represents it perfectly (descriptors and non-descriptors associated with a concept are assumed to be synonymous in the language of the thesaurus)[1].

For the conditional probabilities of each virtual hierarchical node H_C given the concept nodes it comprises, $p(h_c^+|pa(H_C))$, we can use again the previous additive canonical model, because the more relevant are all the concepts contained in the complex concept C associated to H_C, the more clearly this broader concept is appropriate[2]:

$$\forall H_C \in \mathcal{H},\ p(h_c^+|pa(H_C)) = \sum_{C' \in R(pa(H_C))} w(C', H_C). \tag{4.3}$$

$R(pa(H_C)) = \{C' \in Pa(H_C) \,|\, c'^+ \in pa(H_C)\}$ and $w(C', H_C)$ is the weight of the concept C' in H_C, with the weights verifying:

$$w(C', H_C) \geq 0, \text{ and } \sum_{C' \in Pa(H_C)} w(C', H_C) \leq 1.$$

For these weights $w(C', H_C)$, we propose to use uniform weights (there is no prior reason to believe that a concept is more important than another one with respect to the broader concept containing them). Therefore:

$$w(C', H_C) = \frac{1}{|Pa(H_C)|}.$$

Finally, for the conditional probabilities of each concept node given its associated virtual nodes, $p(c^+|\{e_c, h_c\})$, we again propose an OR gate (a concept may

[1] In order to discriminate between concepts having a different number of descriptors that match with the document to be classified, it is preferable not to use a value equal to 1.0.

[2] This strategy is motivated by the common guidelines being used to manually classify documents: we should use the most specific concepts available to bring out the main focus of a document and, if the document covers several specific concepts, then we should use as many specific concepts from different subtrees as required by the content of the document. *However, when several specific concepts are needed that fall within the same subtree structure, the broader concept should be assigned instead.*

become relevant either because of its own lexical information (its descriptor and non-descriptors) or because most of the narrower concepts contained in it become relevant): $\forall C \in \mathcal{C}$,

$$p(c^+|\{e_c, h_c\}) = \begin{cases} 1-(1-w(E_C,C))(1-w(H_C,C)) & \text{if } e_c = e_c^+,\ h_c = h_c^+ \\ w(E_C,C) & \text{if } e_c = e_c^+,\ h_c = h_c^- \\ w(H_C,C) & \text{if } e_c = e_c^-,\ h_c = h_c^+ \\ 0 & \text{if } e_c = e_c^-,\ h_c = h_c^- \end{cases} \tag{4.4}$$

where $w(E_C, C)$ and $w(H_C, C)$ are the weights or importance attributed to the equivalence and hierarchical information, respectively, with $0 \leq w(E_C, C) \leq 1$ and $0 \leq w(H_C, C) \leq 1$.

4.5.3 Inference

The procedure used to classify a given document Q would be as follows: first we instantiate in the network the term nodes corresponding to the words appearing in Q as observed and the remaining term nodes as not observed[1]. Let q be such a configuration of the term nodes in $\mathcal{T}$. Next, we propagate this information through the network and compute the posterior probabilities of the concept nodes, $p(c^+|q)$. Finally, the descriptors associated with the concept nodes having greater posterior probability are used to classify the document.

We can take advantage of both the network topology and the canonical models being considered in order to compute the posterior probabilities of the concept nodes. As all the term nodes are instantiated to either observed or non-observed, then all the descriptor nodes which are parents of a virtual equivalence node are conditionally independent given q. In the same way, the virtual nodes E_C and H_C associated to a concept node C are also conditionally independent given q. Therefore, taking into account that the canonical model for both virtual equivalence nodes and concept nodes is an OR gate, we can compute these probabilities

[1]For that reason the values of the prior probabilities of the term nodes are not important.

as follows [102]:

$$p(e_c^+|q) = 1 - \prod_{D \in Pa(E_C)} \left(1 - w(D,C)p(d^+|q)\right). \tag{4.5}$$

$$p(c^+|q) = 1 - (1 - w(E_C,C)p(e_c^+|q))(1 - w(H_C,C)p(h_c^+|q)). \tag{4.6}$$

The probabilities of the descriptor nodes can be calculated, according to the properties of the additive model being used, as follows [15]:

$$p(d^+|q) = \sum_{T \in Pa(D)} w(T,D)p(t^+|q).$$

As $p(t^+|q) = 1 \; \forall T \in Pa(D) \cap Q$ and $p(t^+|q) = 0 \; \forall T \in Pa(D) \setminus Q$, we obtain:

$$p(d^+|q) = \sum_{T \in Pa(D) \cap Q} w(T,D). \tag{4.7}$$

The computation of the posterior probabilities of the virtual hierarchical nodes is also very simple, using again the properties of the additive canonical model considered:

$$p(h_c^+|q) = \frac{1}{|Pa(H_C)|} \sum_{C' \in Pa(H_C)} p(c'^+|q). \tag{4.8}$$

Therefore, we compute first the posterior probabilities of all the descriptor nodes using (4.7), then the posterior probabilities of the virtual equivalence nodes using (4.5). Next, we can compute in a top-down manner the posterior probabilities of the virtual hierarchical nodes and the concept nodes using (4.8) and (4.6), respectively.

4.5.4 Implementation

Now, let us study in more detail how to implement in an efficient way the proposed model. We start from the term nodes associated with the words appearing in the document to be classified. For each one of them, we accumulate the weights of these term nodes in the descriptor nodes containing them. After this process,

each visited descriptor node D contains the value

$$v[D] = \sum_{T \in Pa(D) \cap Q} w(T, D),$$

which coincides with $p(d^+|q)$, according to (4.7). The posterior probabilities of the non visited descriptor nodes are equal to zero.

Next, starting from each of the visited descriptor nodes, we would visit the virtual equivalence node containing it and compute progressively the product

$$\prod_{D \in Pa(E_C)} (1 - w(D, C)v[D]) .$$

After this step each visited virtual equivalence node contains, according to (4.5), the value

$$v[E_C] = 1 - p(e_c^+|q)$$

(the non visited virtual equivalent nodes have a posterior probability equal to zero).

Finally, we traverse the subgraph induced by the set of visited virtual equivalence nodes and their descendants in a topological ordering (parents before children). If the visited node is a basic concept node C, we directly compute $p(c^+|q)$, by setting

$$v[C] = w(E_C, C)(1 - v[E_C])$$

(because there is no hierarchical information for basic concept nodes). If the visited node is a virtual hierarchical node H_C, we compute its probability by accumulating in $v[H_C]$ the values already computed for its parent concept nodes and dividing by the number of parents, according to (4.8). If the visited node is a complex concept node C, we compute its probability by setting

$$v[C] = 1 - (1 - w(E_C, C))(1 - v[E_C])(1 - w(H_C, C)v[H_C]).$$

The algorithm implementing this process is displayed in Algorithm 3. It can

be easily seen that the complexity of this algorithm is linear in the number of arcs in the graph or, more precisely, linear in the number of arcs of the subgraph induced by the term nodes appearing in the document Q and their descendant nodes. It is worth mentioning that in the actual implementation the Bayesian network is never explicitly constructed; instead, we directly use the BT, NT and USE relationships in the thesaurus, augmented with two inverted file-like structures to store, for each word in the thesaurus, the lists of descriptors and non-descriptors containing it.

4.6 Extending the Basic Model to Cope with Training Information

The model proposed so far does not use training information, in the form of preclassified documents. However, it is quite simple to include this type of information into the Bayesian network model, thus obtaining a supervised classifier. Following with the previously used idea of clearly separating the different sources of information relative to each concept, then we would add a new parent node T_C, called virtual training node, to each concept node C (in addition to those virtual nodes H_C and E_C representing hierarchical and equivalence relationships), representing the information obtained for this concept from the training documents. In other words, this node T_C would contain the posterior probability distribution for the relevance of the concept, predicted by a (probabilistic) supervised classifier. This information would be merged with those obtained from hierarchy and equivalence through an OR gate.

Although, in principle, we could use any supervised classifier able to give a probability distribution as the output, we are going to use a classifier which is particularly coherent with the content of the previous chapter, and with the canonical models being considered: the OR gate Bayesian network classifier, in its multinomial version (noted by "ML" in chapter 3).

Remember that the network structure is fixed, having arcs going from each term node T to the virtual training node T_C if this term appears in training

Algorithm 3 Inference Algorithm for the Basic BN model

for all term node $T \in \mathcal{Q}$ **do**
  for all descriptor node D child of T **do**
    if $v[D]$ exists **then**
      $v[D] \leftarrow v[D] + w(T, D)$
    else
      create $v[D]$
      $v[D] \leftarrow w(T, D)$
    end if
  end for
end for
/* at this point each $v[D]$ contains the value $p(d^+|q)$ for all descriptor node D such that $v[D]$ exists */

for all descriptor node D such that $v[D]$ exists **do**
  E_C : node child of D
  if $v[E_C]$ exists **then**
    $v[E_C] \leftarrow v[E_C]\,(1 - w(D, C)\,v[D])$
  else
    create $v[E_C]$
    $v[E_C] \leftarrow 1 - w(D, C)\,v[D]$
  end if
end for
/* at this point each $v[E_C]$ contains the value $1 - p(e_c^+|q)$ for all node E_C */

insert in a list L the concepts C descendants of all the virtual equivalence nodes E_C such that $v[E_C]$ exists, and their descendants in topological order

for all node $B \in L$ **do**
  if B is a basic concept node C **then**
    $v[C] \leftarrow w(E_C, C)\,(1 - v[E_C])$
  else if B is a virtual hierarchical node H_C **then**
    $v[H_C] \leftarrow 0$
    for all node $G \in Pa(H_C)$ such that $v[G]$ exists **do**
      $v[H_C] \leftarrow v[H_C] + v[G]/k[H_C]$
      /* $k[H_C]$ stores the number of parents of H_C */
    end for
  else
    /* the node is complex */
    $v[C] \leftarrow 1 - (1 - w(E_C, C))\,(1 - v[E_C])\,(1 - w(H_C, C)\,v[H_C])$
  end if
end for
/* at the end of this process each $v[C]$ contains the value $p(c^+|q)$ */

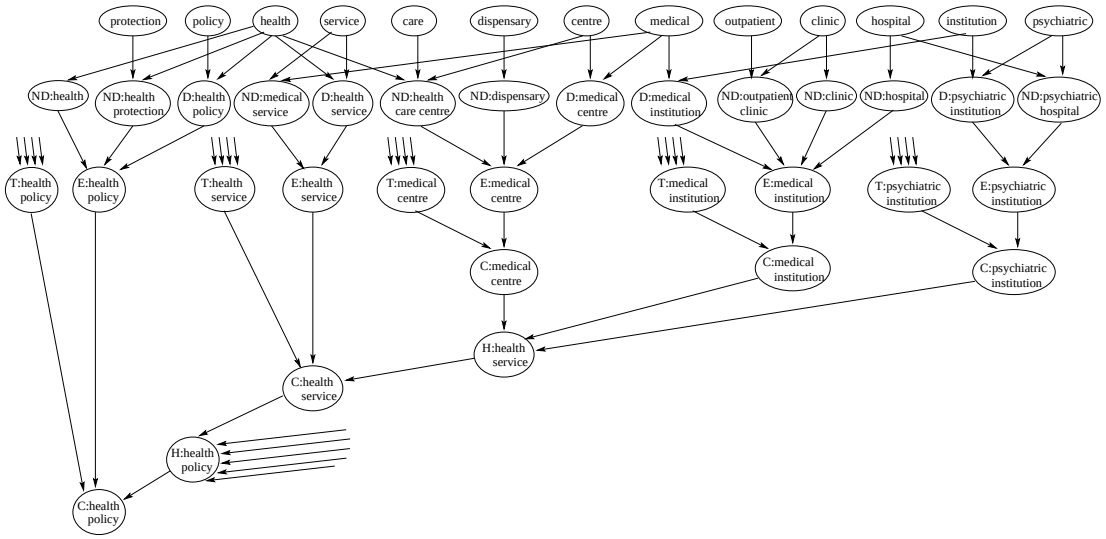

Figure 4.5: Extended Bayesian network to include training information, in the example about *health*.

documents which are associated with the concept C[1]. Figure 4.5 displays the corresponding network for the example about health.

Concerning the numerical information, we include a new parameter $w(T_C, C)$, $0 \leq w(T_C, C) \leq 1$, representing the contribution of the training information to the relevance of the concept C, so that the new conditional distribution of a concept node is

$$\forall C \in \mathcal{C},\ p(c^+|pa(C)) = 1 - \prod_{X_C \in R(pa(C))} (1 - w(X_C, C))\,, \tag{4.9}$$

where X_C represents either E_C, H_C or T_C. The posterior probability of each concept given a document, $p(c^+|q)$, is therefore calculated as:

$$p(c^+|q) = 1-(1-w(E_C, C)p(e_c^+|q))(1-w(H_C, C)p(h_c^+|q))(1-w(T_C, C)p(t_c^+|q)). \tag{4.10}$$

[1]Notice that these terms are no longer restricted to be part of the descriptors in the thesaurus, they are the terms found in the training documents.

On the other hand, the conditional distributions of the new virtual training nodes are defined in the same way as explained in chapter 3 (using the "maximum likelihood" approach).

4.7 Experimental Evaluation

Our experiments have been carried out using a database provided by the Parliament of Andalusia at Spain, containing 7933 parliamentary resolutions manually classified using descriptors from an adapted version of the Eurovoc thesaurus. This specific version contains 5080 descriptors, 6975 non-descriptors and 7120 distinct words (excluding stopwords). The BN representing the thesaurus would therefore contain more than 30000 nodes. The average number of assigned descriptors per document is 3.8 (the number of descriptors assigned to a document ranges from 1 to 14). We have not used the full text of the documents but only a short summary (typically two or three lines of text). In our experiments we always use stemming (provided by Porter's algorithm for Spanish language implemented in the Snowball package [104]) and stopword removal.

The evaluation takes into account that our aim is not a complete but only a partial automation of the classification process, showing to the user an ordered list of the most probable descriptors[1] Then, as performance measures, and following our analysis made in section 4.3.5 we have firstly selected the typical measure used in multilabel categorization problems: precision/recall breakeven point, which will be computed in micro-average and macro-average. The two presented IR-style measures will also be used too: the F_1 *measure* at a five document level (i.e. F_1@5, the F_1 obtained by assuming that the system assigns to each document the *five* most probable descriptors), and the *average 11-point precision*. As in the case of the breakeven point, we shall compute the micro and macro averages of the F_1 measure. In all the measures, a higher value means a better performance of the model, and all of them are real numbers on the interval $[0, 1]$.

[1] We are therefore using an instance of the so-called category-ranking classifiers [119].

4.7.1 Experiments without Using Training Documents

In order to assess the quality of the proposed Bayesian network based model without using training data, we have also experimentally compared it with the two simple VSM models presented on section 4.3.4, used as basic benchmark methods. Let us recall that the first one (VSM) ranks concepts for a document based on word matching between the document and the lexical information associated to the concepts in the thesaurus (each concept is indexed using the words appearing in the descriptor and non-descriptors which are associated with it), so hierarchical information is neglected. The second one, the hierarchical information (HVSM), is based on the idea that the meaning of a concept in the thesaurus is a specialization of the meaning of the broader concepts containing it[1] (all the words appearing in the descriptors and non-descriptors of the broader concepts of a given concept are also used to index the "document" associated with this concept).

Several combinations of parameters have been tested for our Bayesian network-based model (BN). In particular, the parameters chosen to be variable have been the weights $w(H_C, C)$ and $w(D, C)$. As stated in section 4.5.2, we have chosen high values for the weight $w(D, C)$ (0.8 and 0.9), together with the value 1.0. In order to test the value of the hierarchical information, we have selected several high values (0.8, 0.9 and 1.0) and a low value (0.0). On the other hand, the value of the weight of the equivalences $w(E_c, C)$ has been fixed to 1.0. Then, a value $BN, 0.9, 0.8$ in the table 4.2 means the Bayesian network model with $w(D, C) = 0.9$ and $w(H_C, C) = 0.8$.

With respect to the efficiency of the inference process, all the 7933 resolutions were classified in around 10 seconds on a computer equipped with an Intel Core2 duo 2GHz processor.

The main conclusion that may be obtained from these experiments is that the Bayesian network approach is useful in this classification problem, since it always provides much better results than both the simple and hierarchical vector space models. The model performance is in general quite acceptable, taking into

[1] In the language of our Bayesian network model, these broader concepts would be the descendants of the concept being considered.

account that no training documents have been used. Concerning the vector space model, in this case the use of the hierarchical information is self-defeating and produces results worse than those of the simple VSM[1].

The parameters chosen show better performance for a weight of the descriptors near 1.0, than the 1.0 itself. Although the usage of a hierarchy weight distinct to 0.0 does not strongly boost the results, it performs little improvements in the measures, specially in the average precision.

Models	Micro BEP	Macr BEP	Av. prec.	Micro F1@5	Macro F1@5
BN, 0.8, 0	0.26244	0.20394	0.29967	0.30811	0.17661
BN, 0.9, 0	**0.28241**	0.20234	0.30700	**0.31419**	0.18419
BN, 0.8, 0.8	0.26068	0.21208	0.30500	0.30845	0.17521
BN, 0.9, 0.9	**0.26881**	**0.20903**	**0.31321**	**0.31473**	**0.18433**
BN, 0.9, 1.0	0.26636	**0.20880**	**0.31261**	0.31381	0.18265
BN, 1.0, 1.0	0.25584	0.20768	0.27870	0.30963	**0.18865**
VSM	0.15127	0.18772	0.18061	0.20839	0.17016
HVSM	0.13326	0.17579	0.17151	0.20052	0.14587

Table 4.2: Performance measures for the experiments without using training documents (the two best values for each column are marked in boldface).

4.7.2 Experiments Using Training Documents

In this section we shall evaluate the results obtained by the model using training documents. We also want to evaluate how the system improves its performance as more training data are available. In all the computed measures through the experiments, we shall use 5-folds cross-validation over the same 5 partitions of the collection. The presented measures, then, will be the average of the five obtained values. The evaluation will be carried out with the same five measures chosen for the previous experimentation, in order to make both comparable.

In the first part of the experimentation, the supervised approach of our Bayesian network will be compared against four pure supervised approaches to multilabel classification. Concretely, we will use the multinomial Naïve Bayes

[1]This contrasts with the results obtained in [2] in the context of hierarchical classification of documents into web directories, where the hierarchical VSM generally outperformed the simple VSM.

model [75; 85], the Rocchio classifier [59], Support Vector Machines (SVM) [60] and a standalone OR gate classifier model[1], constructed using information only from training documents (and not taking into consideration neither the thesaurus lexical information nor its hierarchical structure)[2].

The first set of experiments, whose results are displayed in table 4.3, compares the four supervised approaches with the model using training documents, tuning some parameters. As stated before, $w(D, C)$ should have a high value, near 1.0. This parameter will be fixed to 0.9 (a value which provides good results on the previous experimentation). On the other hand, the weight of the training information, $w(T_C, C)$, will be high, and also fixed (to 1.0 in this case). Therefore, the two free tunable parameters we shall consider in the model will be the weight of the hierarchy, $w(H_C, C)$, and the weight of the equivalence relationships, $w(E_C, C)$. In table 4.3, the supervised version of our Bayesian network model will be noted as "SBN a, b", where a will be the weight $w(E_c, C)$ and b will be $w(H_C, C)$. From a certain viewpoint, we want to study the contribution of these two sources of information (hierarchical and lexical) to the baseline model (the standalone OR gate classifier). This leads us to the two following questions. Does information from the terms of the thesaurus help in the supervised case? And the second one, does information from the hierarchical relationships of the thesaurus helps now?

These experiments show up that adding hierarchical information ("Sup. BN, 0.0, X") to the OR gate model clearly improves the classification results. Moreover, adding textual information ("Sup. BN, X, 0.0") without hierarchical information also boots classification results. In this case, the hierarchy added to the lexical information of the thesaurus does not make a significant advance, but it improves the results, being the "Sup. BN, 0.5, 0.1" and the "Sup. BN, 0.8, 0.1" the two best performing configurations we have tested.

The results in Table 4.3 show that our Bayesian network model systematically obtains better results than two classical supervised classifiers (Rocchio and Naïve Bayes) and one 'uninformed' version of itself (standalone OR gate), and even

[1]The one presented in chapter 3, with the "maximum likelihood" estimation.

[2]In all the cases we used our own implementations of these algorithms, except in the case of SVM, where the software package SVM^{light} [62] was used.

Models	Micro BEP	Macr BEP	Av. prec.	Micro F1@5	Macro F1@5
Naïve Bayes	0.42924	0.17787	0.61840	0.50050	0.20322
Rocchio	0.34158	0.35796	0.43516	0.40527	0.33980
OR gate	0.40338	0.44855	0.56236	0.41367	0.24629
SVM	**0.63972**	0.47890	0.69695	0.57268	**0.40841**
SBN 0.0, 0.9	0.54825	0.43361	0.66834	0.54066	0.33414
SBN 0.0, 0.8	0.55191	0.43388	0.67149	0.54294	0.33781
SBN 0.0, 0.5	0.55617	0.43269	0.67571	0.54578	0.34088
SBN 0.0, 0.1	0.55735	0.43282	0.67761	0.54652	0.34228
SBN 0.9, 0.0	0.55294	0.47207	0.65998	0.56940	0.36761
SBN 0.8, 0.0	0.57936	0.47820	0.68185	**0.58163**	**0.38589**
SBN 0.5, 0.0	**0.58372**	**0.48497**	**0.70176**	0.57875	0.38009
SBN 0.1, 0.0	0.56229	0.46171	0.68715	0.55390	0.35123
SBN 0.8, 0.1	0.57887	0.47809	0.68187	**0.58144**	**0.38610**
SBN 0.5, 0.1	**0.58343**	**0.48487**	**0.70197**	**0.57887**	0.38146
SBN 0.5, 0.5	0.58285	**0.48716**	**0.70096**	0.57859	0.37868
SBN 0.8, 0.8	0.56801	0.47946	0.67358	0.57508	0.37300
SBN 0.9, 0.9	0.53963	0.47200	0.64957	0.56278	0.35742
SBN 1.0, 1.0	0.49084	0.45875	0.59042	0.53235	0.32173

Table 4.3: Performance measures for the experiments using training documents (the three best values for each column are marked in boldface).

outperforms SVM in some cases. Nevertheless, it should be noticed that in any case the performance measures obtained are not very high. This can be explained if we consider the fact that performance decreases as the number of categories in the problem being considered increases [5; 134], and in our case the number of categories is quite high (in the order of thousands). However, as we explained earlier, our goal is not to replace the human experts but to help them, by providing an ordered list where the correct descriptors can be found in the first positions in the list. In order to show that this is indeed the case, we have carried out another experiment to compute the average recall values[1] obtained by the different classifiers when we display to the user the n top-ranked categories, for $n = 5$, 7, 10, 12, 15, 17 and 20. The results are displayed in Figure 4.6.

[1]The proportion of correctly assigned categories with respect to the total number of true categories associated with each document

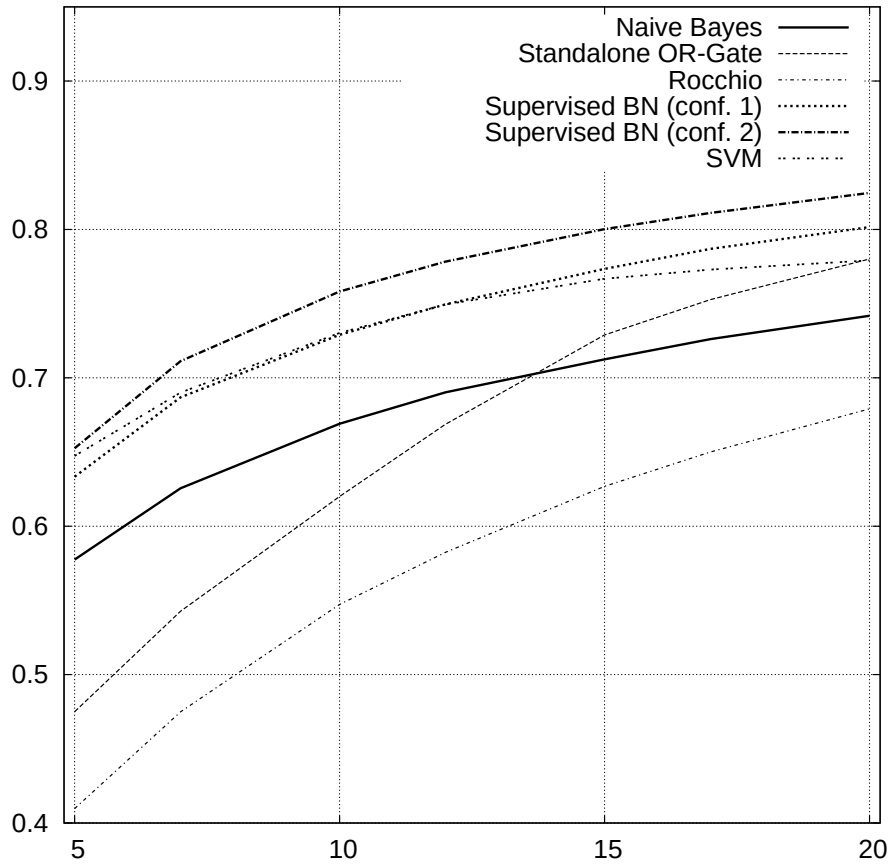

Figure 4.6: Microaveraged recall values computed for incremental number of displayed categories

We can observe in the figure that one of our models finds 65% of the true categories among the first five categories in the list, 75% among the first ten and 80% among the first fifteen (from a list of 5080 possible categories). We believe that any human indexer would consider useful a system having these characteristics.

The second part of the experimentation will test the classification models in an environment where not all the training data is available. In these experiments, for a same test partition, all the classifiers will be trained with the 10%, 20%, ..., 100% of the training data, in order to study if our Bayesian network model keeps its advantage with the classical models, and if it needs less data to achieve

a good performance. We have selected, for comparison, two of the best performing parameter configurations, "SBN 0.5, 0.1" and "SBN 0.8, 0.1" which will be referred in the experiments as configuration 1 and 2, respectively.

For each measure (micro and macro averaged BEP, average precision and micro and macro averaged F_1 at five), we have obtained a graph, with the values of the measure at those training set percentages. In all cases, the results are also averaged over the five test partitions. The results are displayed in figures 4.7, 4.8, 4.9, 4.10 and 4.11.

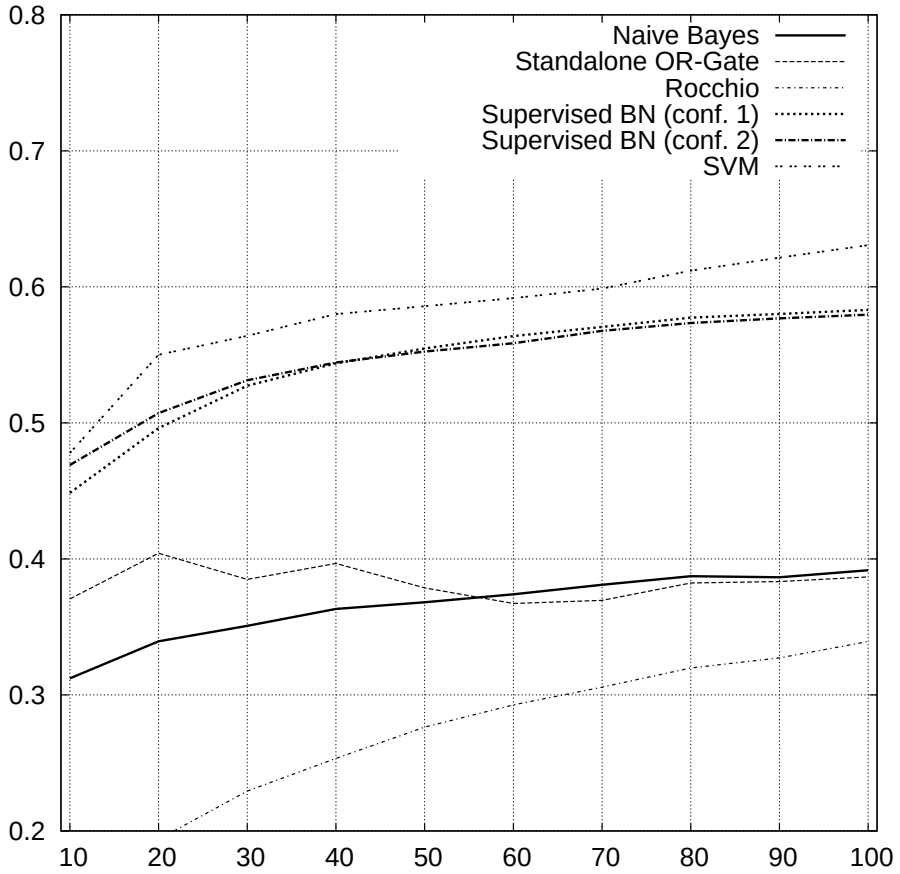

Figure 4.7: Microaveraged breakeven point computed for incremental percentage of training data.

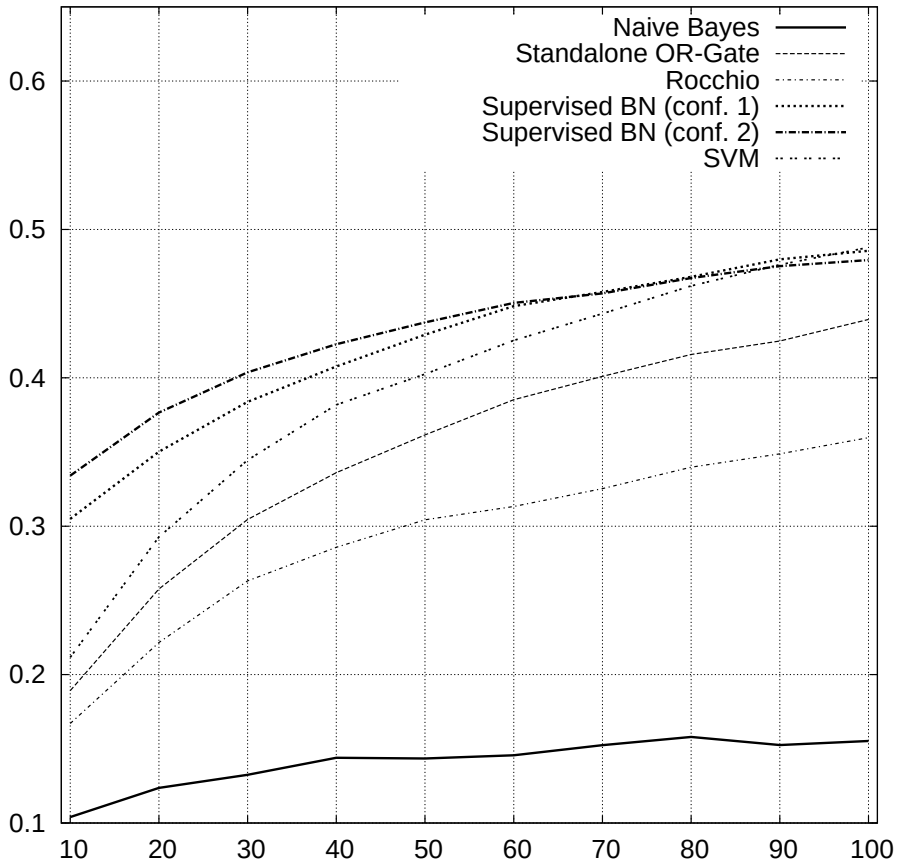

Figure 4.8: Macroaveraged breakeven point computed for incremental percentage of training data.

The results speak for themselves: the Bayesian network model shows a great difference with two of the classical supervised approaches (Rocchio and naive Bayes) and with the OR gate model, in all the cases; in particular, when few

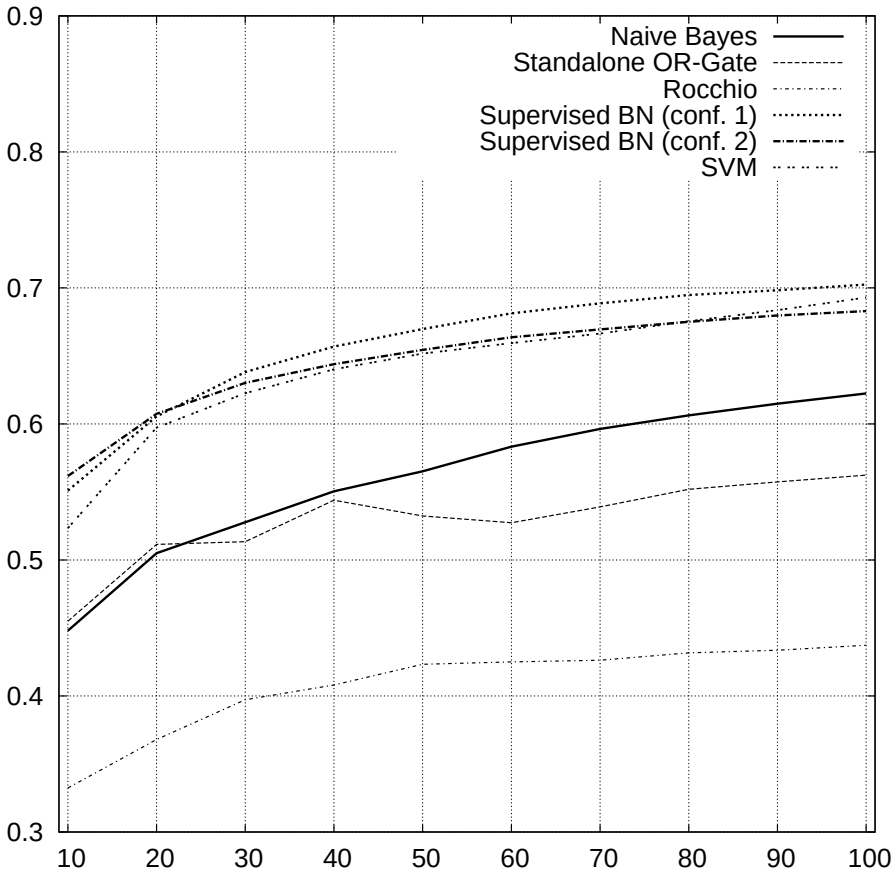

Figure 4.9: Average precision at 11 standard recall points computed for incremental percentage of training data.

training information is available, our model also outperforms SVM in most of the cases. Our model also tends to stabilize before and to obtain results close to the maximum in an early stage of the curve.

4.8 Concluding Remarks

4.8.1 Conclusions

We have developed a Bayesian network-based model for hierarchical classification of documents from a thesaurus. The experimental results obtained using a large

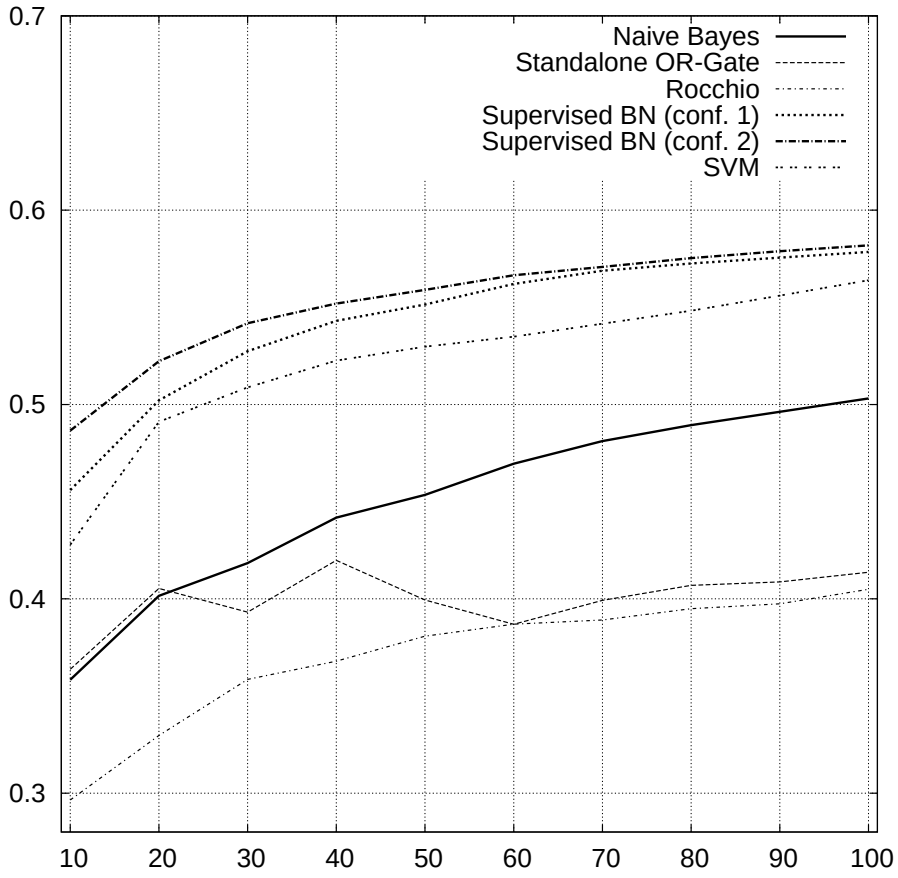

Figure 4.10: Micro F_1 at five computed for incremental percentage of training data.

set of parliamentary resolutions from the Parliament of Andalusia and the Eurovoc thesaurus are encouraging: the model without training clearly outperforms the two simple benchmark methods considered; by integrating the initial model within a more general scheme where training data, in the form of preclassified documents, may also be used, we have also outperformed standard text classification algorithms, as Rocchio and Naive Bayes, obtaining results comparable to those of Support Vector Machines.

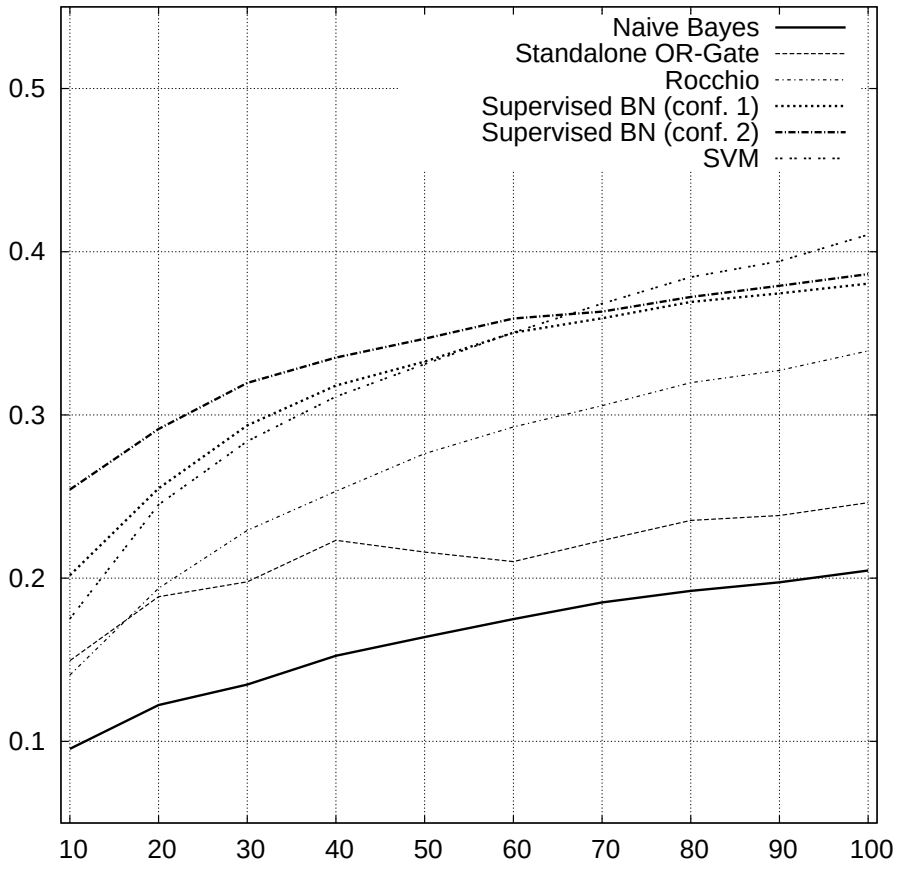

Figure 4.11: Macro F_1 at five computed for incremental percentage of training data.

4.8.2 Future Work

For future research, we are planning to improve the initial model in two different ways: first, by considering the *context* of the terms/descriptors appearing in a document. The idea is to avoid assigning to a document a descriptor whose appearance may be incidental or their meaning within the document being quite different from the intended meaning within the thesaurus. Second, by taking also into account the associative relationships between descriptors in the thesaurus.

This initial model could also be combined with other supervised text classifiers different from the OR gate classifier. Perhaps the relative weights of the lexical

information in the thesaurus (descriptors and non-descriptors) should depend on the amount of training data (the more training data, the less influence of the lexical information in the thesaurus). More generally, instead of fixing manually all the parameters, i.e. the weights of the different canonical (additive and OR) probability models being used, it would be interesting to try to estimate or learn these parameters from the training data.

The Bayesian network model proposed in this chapter is focused on classification using descriptors of a thesaurus. However, it could also be used in other classification problems where the different classes have associated some kind of descriptive text (which would play the role of descriptors), for example the problem of classifying documents into hierarchical web directories. Moreover, the model could also be used with a minor modification in hierarchical text classification problems, provided that the document can be associated with internal categories (and not only with the leaves categories): by removing the virtual equivalent nodes (as well as descriptor nodes). We plan to test our model in these kinds of problems, as well as with other thesauri larger than Eurovoc, as AGROVOC or MeSH.

From a methodological point of view, we are also interested in building of several test collections. As we presented before, official thesauri are used in many institutions, and it would be interesting to collect, for a certain thesaurus, a considerable amount of manually indexed documents into a test collection, together with the edition of the thesaurus used to classify them. This could help researchers developing new models, and make this model comparable with new propositions.

Besides, the question of what is the suitable way of evaluating this task, taking into acount the hierarchy remains as an open question. As explained before, traditional precision-recall metrics (and derivatives) do not consider the hierarchical relationships among classes. A more precise measure should take this structure into account, not penalizing a result if it is related with the true value on its ancestral line. We think that more work could be done in this task.

Chapter 5

Structured Document Categorization Using Bayesian Networks

5.1 Introduction

In this chapter we deal with a subproblem of supervised Document Categorization, which we shall call Structured Document Categorization. Because this is not a standard terminology, we shall make a brief introduction to what we understand as "structured".

As stated before, in chapter 1, the "classic" Document Categorization problem consisted of finding a reasonable labeling for a set of flat text documents, called the test corpus. To make this labeling we are provided of another corpus, the training corpus, whose examples are already labeled (with binary labels, multiclass labels, or even multilabeled). The classic methods exploit the content of the labeled documents, in order to find a good classifier which assigns the label to unlabeled documents, as a function of the content itself (the terms). The classic problem is very well studied [118], and the state-of-the-art methods often outperform humans doing the same task, for small corpora. These documents are what we have called "flat text documents" (or simply "flat documents") in previous chapters. The term "flat" is used here as "absence of any kind of organization"

(apart from the organization given by the documents themselves) or just as an antonym of "structured".

Thus, unlike flat documents, structured documents add some kind of organization to the text itself. Nowadays, with the advent of the Internet and the explosion of information several ways to organize such amount of information have arisen, and gain credibility. These methods to add some organization are very commonly used, and result in document models which are more realistic than simply independent flat text document. We enumerate three of the main methods to organize text document collections:

1. The documents can be allocated in a **hierarchical directory**, in order to make their access easier.

2. The information in each document can be **divided and organized, internally**, making a document a non-atomic entity, and allowing the user to access to any of its parts.

3. The corpus of documents can allow to make **explicit relationships (links) among documents**, indicating that the content of the linked document is somewhat related to the document that links it.

The first method does not allow structured information itself. Thus, it keeps documents organized in a set of hierarchically organized categories, but the problem of classification in such a environment is mainly devoted to develop a mathematical model of the explicit relationships among categories. In fact, the problem of document categorization in a hierarchy of classes is a very active field of research, and the model developed by us in chapter 4 is, from a certain point of view, a model of such a kind.

The second and the third methods modify explicitly the content of the corpus, adding to the textual information some kind of meta-information. These two final methods to represent documents are those which we refer to when we speak of Structured Document Categorization.

Roughly speaking, the *problem of Structured Document Categorization* can be defined as a problem of text document classification, in which a certain *internal structure* in the documents exists (inside each document, independently, as in the

second case of non-atomic documents, or with explicit relationships -links- among documents).

As a nomenclature convention, from this point, we shall refer to an internally structured document simply as a "structured document", knowing that the structure underlying on the third kind of documents is inherent to the whole set of documents (a more precise terminology for the third kind of organization would be "structured corpus"). Thus, the third problem will be referenced in the following sections as *Link-based Document Classification*, in order to be distinguished of the previous one.

The outline of this chapter is presented as follows: to begin with, we firstly deal with the problem of (Internally) Structured Document Categorization. We present this subproblem, studying first a language to define internal document structure in section 5.2. After that, we define the problem of internally Structured Document Categorization (5.3.1) and we make a review of the existing methods for it (section 5.3.2). On section 5.4 we present our methods for classification (which will essentially consist of reducing this problem to one of the classic "flat classification", using a special setting), and after that we include a section of experimentation (section 5.5) of this methodology with a suitable corpus. Partial conclusions of this part are shown at section 5.6.

Secondly, in section 5.7 the problem we are going to deal with is the one where we have a set of linked documents. The problem will be presented, and we shall review some previous work on this Link-based Classification in sections 5.8.1 and 5.8.2, respectively. On this subproblem, we shall present two different models, both based on Bayesian networks. The first one, developed in section 5.9, will be a very simple network, with fixed structure, designed to make multiclass classification. Conversely, the second one (presented in 5.12) will be a model where the structure of the Bayesian network will be learnt directly from data, and it will be used to cope with multilabeled corpus. Each model will be tested and evaluated with a suitable corpus (in 5.10 and 5.13, respectively), and results will be presented. Finally, some conclusions about every one of the two subproblems will be stated in sections 5.11 (for the multiclass model) and 5.14 (for the multilabel model).

5.2 Structured Documents

5.2.1 What is a Structured Document?

The kind of textual information considered until this point has been composed of independent and atomic units called documents. These units can represent literary works, scientific articles, annotated multimedia files, etc. If we think of the concept of document, there are multiple examples where, even being possibly considerable atomic, it is more natural to treat a document as a set of different parts. We can think in the following examples:

- A book can be divided in chapters (if it is a literary work). In the case of dealing with a play (theatrical literature) we shall surely have more divisions, for instance, acts, scenes, lines, speeches, etc.
- A scientific paper often starts first with a title, then with an abstract followed by a set of sections, each one usually divided into several subsections (and so on), and finally, the set of bibliographical references.
- The written transcriptions of a trial, which normally follows a structure that, even not being rigid on the order of the different parts, has several compulsory divisions that form it.

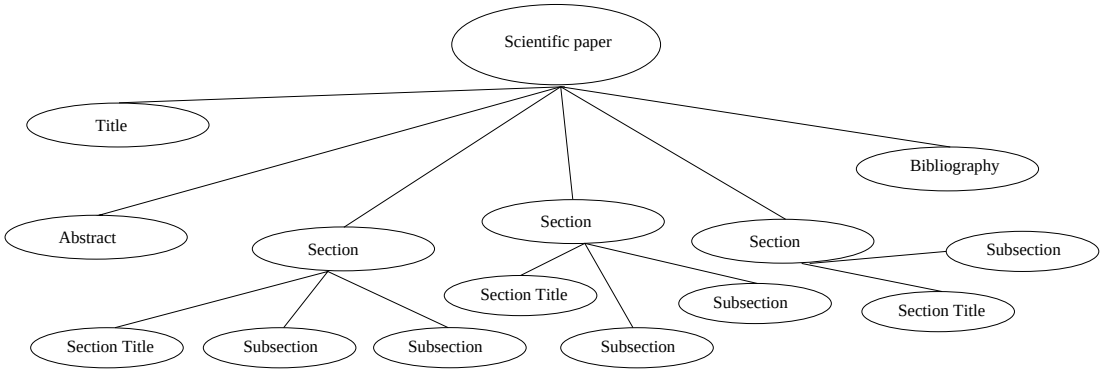

Figure 5.1: Example of a structured document.

This kind of documents presents a different panorama to the one described on chapter 1. Then, a corpus was a synonym for a set of documents $\mathcal{D} = \{D_1, \ldots, D_m\}$, but now we have a set of *structural units* $\mathcal{U} = \{U_1, \ldots, U_p\}$. One structural unit is a continuous[1] fragment of a flat text document (the structural document considered as a whole, with no structural divisions).

We can preserve the link among units and documents, introducing the membership relation $U_i \in D_j$, meaning that a structural unit U_i belongs (is a fragment of) to one document D_j. Obviously, it holds that $\forall U_i \in \mathcal{U},\ \exists!\, D_j : U_i \in D_j$ (each structural unit belongs to only one document).

Moreover, the set of structural units should form a tree, or a forest [44], that is to say, we can define for each structural unit, the map container,

$$\text{container} : \mathcal{U} \longrightarrow \mathcal{U} \cup \{\emptyset\},$$

which represents the structural unit that contains one given. It should be verified, $\forall U_i, U_j, U_k \in \mathcal{U}\ \wedge \text{container}(U_i) \neq \emptyset$, that:

(1) $\text{container}\overbrace{\circ \cdots \circ}^{k}\text{container}(U_i) \neq U_i, \forall U_i \in \mathcal{U}, \forall k = 1, 2, \ldots$ (non existence of cycles), and,

(2) $\text{container}(U_j) = U_k \Rightarrow \exists!\, D_l : U_j, U_k \in D_l$ (all the units of the same tree belong to only one document).

Finally, every structural unit can contain text, other units (of which it its their container), both of them, or none (if the unit is empty, and is placed, for example, on a specific part of the document). Further specifications on what can be the type of the content of a unit can be stated with the languages for describing structured documents.

Each structured unit is associated to a set of tags, $\mathcal{G} = \{g_1, \ldots, g_q\}$ (each unit has one tag) which, in some occasions, add some semantic meaning to the structural division (for example, for scientific papers, we could have $\mathcal{G} = \{$"title", "author", "abstract", "section", $\ldots\}$.

[1] That is to say, if we consider the text of the document as a succession of words, if a unit contains two chosen words, it should also contain all the words between the two first.

Note that the concept of structure that we have defined is flexible. In fact, *structured information is information that is analyzed* [41]. This means that building a structured document for some given flat documents implies choosing an ideal set of tags, $\mathcal{G}$ which should be informative of the kind of unit it represents. If we choose a set $\mathcal{G}$, only with criteria based on the final presentation of the document (font size, font aspect, etc), we could probably obtain very little information of the structure. On the other hand, choosing where structural units are placed (that is to say, dividing those document in a set $\mathcal{U}$) is not an easy task. Not all the divisions are useful, neither they all help to organize better a set of documents. For this case, it is clear that *how you divide up your data matters* [41].

One example of the relationships among structural units of a structured document can be visualized in figure 5.1 (the text has been omitted for clarity reasons).

5.2.2 Languages for Structured Documents

Now we deal with the practical definition of a structured document. It is very well known that the de facto standard to define structured documents is the *Extensible Markup Language* (XML) language. This language was created by the World Wide Web Consortium (W3C, an international organism that watches over the improvement of the Internet), using the previous specifications of the SGML (a language used previously for the same task). The first version of XML was announced on 1998 and it is now widely used. Its syntax is very easy, independent of the platform used, and it completely supports internationalization.

Moreover, the *Extensive* term (for which the first X of XML stands for) means that XML has not a restricted set of tags (that is to say, you can use any set $\mathcal{G}$ for any document collection).

The syntax of XML is very simple. After one header including the XML version number and the encoding of the text used (UTF-8, ISO-8859-2, etc), the proper content of the file is presented. We shall use a markup of the type `<tag>` (to start a structural unit of type "tag"), and `</tag>` (to finish that structural unit). The only imposed restrictions are the following:

- Every tag `<tag>` that is opened, should be closed with its corresponding `</tag>` mark.
- The XML elements should not overlap. If one unit of type `<tag1>` contains another unit `<tag2>`, the correct syntax will be: `<tag1>` ... `<tag2>` ... `</tag2>` ... `</tag1>`. Note that the first unit is closed after the nested one is finished, not the opposite.
- The whole XML document should be contained inside only one unit, called the root element.

If an XML document verifies this restriction we shall say that it is *well formed XML*. Some of the methodologies that we shall present in section 5.4 will require this correct setting, and all of the software tools used to process XML documents also need this.

5.3 Structured Document Categorization

5.3.1 Statement of the Problem. Taxonomy of Models

Given a set of training documents $\mathcal{D}$ (which can be seen as a set of structural units $\mathcal{U}$ where each unit belongs to a document, and verifies all the rules stated in section 5.2), the problem of Structured Document classification consists in finding one classifier f which, being based on the training data, will be able to categorize a new document D of the same nature (formed by structural units). That is to say, finding a classifier able to assign a document D one or several categories $f(D)$ (depending on the kind of the problem chosen).

In fact, and as in the case of flat document classification, the problem can be *binary* (the classifier can choose between one class and its negation), *multiclass* (there are more than two categories to be assigned, but only one is chosen), and *multilabel* (a subset of categories is assigned to every new document to be categorized).

As we shall see in the next section (5.3.2,containing a review of models), the two different approaches to this problem are the following:

1. A **reduction** of this problem to a flat Document Classification one, where the structural information is used to build a new set of features. This approach is easy because all the previous development in this field can be used, and only some transformations on the documents need to be done.

2. A **combination** of flat classifiers, launched on the set of leafs of a structural tree. This is a rather computationally expensive approach, because it needs more classifiers to be trained, and executed (usually one for each structural unit, or for each structural unit with a decent size).

On the first type we find the *tagging* approach (each term of the document is "tagged" with a prefix indicating structural information), and all the approaches that add new features built only with structural information. For the second type, we have the *splitting* and *stacking* approaches. In the former, a separate model is built for each distinct document part and then they are combined in a way that is natural to the underlying classification algorithm. In the latter approach, predictions based on different structural components are combined by a meta classifier built on these predictions.

In this dissertation, we shall not be using any of the two *meta*-classification approaches (the *combination* methods of the second point). Instead of that, in section 5.4, we shall review some *reduction* procedures, trying to build new document representations, and applying some classic classifiers, or some of those presented in chapter 3. We shall also present new reduction approaches, and several variations of the classic ones, along with an experimental evaluation of them.

5.3.2 Previous Works on Structured Document Categorization

This area is a relatively new and unexploited field of research. The first work on classification of documents where the structure was used [138] dates from 2000 and it presents a tree structure to model structured documents. After that, each term is tagged with the path to the node (building then a new feature set), applying afterwards a Naïve Bayes classifier (the first application of the *tagging*

method). Another idea presented on this work was to train a different classifier for each different path, and combine them. This last approach resulted in being worse than the *tagging* one.

The splitting approach is used in [36], where a SVM with a Fisher kernel is applied to the leaf nodes, on the top of a Naïve Bayes classifier, with great results over the flat baselines.

In [9] a deep review of this field is presented, and a good set of experiments is done with several datasets of different nature.

A very specific case of Structured Document Categorization is e-mail classification, where the structure is rather fixed (an e-mail is composed of a sender address, a set of recipients, the subject, the header, the body and sometimes the attached files). On this subproblem, some works have treated it as a Structured Document Categorization problem with two different approaches: training of a different classifier for each one of the parts of the e-mail (as in [65]), or adding the structural features to the body of the e-mail [10] (that is to say, the problem of Structured Document Categorization is thus reduced to a classic Document Categorization problem).

5.4 Development of Several Structured Document Reduction Methods

In this section we deal with the problem of finding new structured document representations on the space of flat documents (i.e., given a structured document, we obtain a transformed flat document, where some of the structural information has been used to build new features). These methods, along with the results presented, are the same we showed in our INEX'07 paper [16].

It is obvious then, that these methods are independent of the classifier used, and so, it is possible to make all possible combinations between reduction procedures and classifiers, which gives us a huge amount of categorization approaches.

In order to clearly present all the reduction procedures, we shall use the small XML document displayed in figure 5.2 (the beginning of "El Quijote") as a guiding example of the transformations applied to the XML.

```
<book>
 <title>El ingenioso hidalgo Don Quijote de
 la Mancha</title>
 <author>Miguel de Cervantes Saavedra</author>
 <contents>
   <chapter>Uno</chapter>
    <text>En un lugar de La Mancha de cuyo nombre
     no quiero acordarme...</text>
 </contents>
</book>
```

Figure 5.2: "Quijote", XML Fragment used for examples, with header removed.

We now explain the different approaches to map structural files into flat ones.

5.4.1 Method 1: "Only text"

This is the naïve approach. It consists of removing all the structural marks from the XML file, obtaining a plain text file. Used with the previous example, we get the document displayed in figure 5.3.

```
El ingenioso hidalgo Don Quijote de la Mancha Miguel
de Cervantes Saavedra Uno En un lugar de La Mancha de
cuyo nombre no quiero acordarme...
```

Figure 5.3: "Quijote", with "only text" approach.

This method should be taken as a *baseline*. Thus, removing all the structural information, we have a starting point whose classification accuracy should be improved using more advanced representations.

5.4.2 Method 2: "Adding"

This method adds structural features to the document, separated from the textual features. That is to say, structural units are introduced as "additional terms" into the document. This is basically a natural extension of the procedure presented in [10]. Our method includes the possibility of adding structural features at a certain depth.

Thus, we can see a structural unit in an "local" way (an unit is characterized only by its tag), or we can consider the influence of the list of units to the root element, until a certain depth level (an unit is characterized by its tag, followed by the tag of the container, and so on). Using the previous example, the `text` unit can be seen standalone ("adding_1", with depth = 1), `contents_text` ("adding_2", depth = 2) or `book_contents_text` ("adding_0", maximum depth value).

We show in figure 5.4 the transformed flat document of the example document using "adding" with depth = 2. Leading underscores are used to distinguish between textual terms and terms representing structural marks. In order to understand better this procedure, we have made, in figure 5.5, the same procedure using "adding" with depth = 1. Note that, if the list of successive containers of a unit has a length less than the depth of the method adding, we just add the set of containers until the root is found.

```
_book _book_title El ingenioso hidalgo Don Quijote
de la Mancha _book_author Miguel de Cervantes Saavedra
_book_contents _contents_chapter Uno _contents_text
En un lugar de La Mancha de cuyo nombre no quiero
acordarme...
```

Figure 5.4: "Quijote", with "adding_2".

```
_book _title El ingenioso hidalgo Don Quijote de la
Mancha _author Miguel de Cervantes Saavedra _contents
_chapter Uno _text En un lugar de La Mancha de cuyo
nombre no quiero acordarme...
```

Figure 5.5: "Quijote", with "adding_1".

5.4.3 Method 3: "Tagging"

This approach is the same as the one described in [9; 138], and also named "tagging". It considers that two occurrences of the same term are different if they appear inside two different structural units. To model this, terms are "tagged" with a representation of the structural unit they appear in. This can be easily simulated appending a prefix to the term, representing its container.

Again, we add the possibility of experimenting at different depth levels, as we did in the method "adding".

The only drawback of this procedure is that the data preprocessed by this method can be very sparse, and then, a very large lexicon could be built from medium sized collections.

For our example document this method, with depth = 1, obtains the flat document displayed in figure 5.6.

```
title_El title_ingenioso title_hidalgo title_Don title_Quijote
title_de title_la title_Mancha author_Miguel author_de
author_Cervantes author_Saavedra chapter_Uno text_En text_un
text_lugar text_de text_La text_Mancha text_de text_cuyo
text_nombre text_no text_quiero text_acordarme...
```

Figure 5.6: "Quijote", with "tagging_1".

5.4.4 Method 4: "No text"

This method tries to unveil the categorization power using only structural units, processed in the same way as in the "adding" method. Roughly speaking, it is equivalent to "adding" and then removing textual terms. In figure 5.7 we can see the "notext_0" processing of the previous example.

```
_book _book_title _book_author _book_contents
_book_contents_chapter _book_contents_text
```

Figure 5.7: "Quijote", with "notext_0".

5.4.5 Method 5: "Text replication"

The previous methods deal with a structured collection, having no previous knowledge about it. That is to say, they do not take into account the kind of mark, in order to select one action or another. This approach assigns an integer value to each tag, proportional to its informative content for categorization (the higher the value, the more informative). This value is used to *replicate* terms, multiplying their frequencies in a mark by that factor. Notice that only values for structural marks directly containing terms must be supplied.

In the previous example, suppose we assign the following set of replication values:

```
title: 1, author: 0, chapter: 0, text: 2
```

Note that a value of 0 indicates that the terms in that unit will be removed. The resulting text is displayed in figure 5.8.

This method is very flexible, and it generalizes several ones, as the "only text" approach (one may select 1 for all replication values). The method consisting of just selecting text from certain tags can be simulated here using 1 and 0 replication values if the text into an unit is to be considered or not, respectively.

```
El ingenioso hidalgo Don Quijote de la Mancha En En un
un lugar lugar de de La La Mancha Mancha de de cuyo cuyo
nombre nombre no no quiero quiero acordarme acordarme...
```

Figure 5.8: "Quijote", with "replication" method, using values proposed before.

The main drawback of "text replication" is that we need some experience with the collection, in order to build the table of replication values before processing the files.

5.5 Experiments with a Structured Document Corpus

We present here an experimental evaluation carried out with the Wikipedia XML corpus [37], on the background of the INEX'07 XML Document Mining track. The results presented here are basically those obtained in our participation on the track [16].

The INEX'07 Document Mining corpus is composed of 96611 documents extracted from the Wikipedia XML Corpus [37] with a 50% training/test split. The number of categories is 21, corresponding to several portals (for example Portal:Law, and Portal:Trains) of the Wikipedia. The corpus is relatively large (720 MB) compared with the standards in Text Categorization, and has over 4 million of structural units. Besides, the corpus is a single-label one (that is to say, each document is assigned to only one category).

Previous to the classification, and in order to select the best combinations of classifiers and representations, we have carried out some experiments using only the training set, by means of cross-validation (dividing the training set into 5 parts). The selected evaluation measures are the microaverage and macroaverage breakeven point (for *soft categorization*) and microaverage and macroaverage F1

(for *hard categorization*). In every case, the "only text" representation will be used as a baseline to compare results among different alternatives.

On the other hand, the different classifiers used will be the Naive Bayes in its multinomial version, and the OR Gate in the two different versions presented on chapter 2: with maximum likelihood estimation (ML) and with the term interaction formula (TI).

A cautious exploration of the collection gave us enough information to propose a set of different replication values (obtained intuitively). Table 5.1 displays the replication values used in the experiments with the "text replication" approach, for the different tags. Tags with unspecified replication values are always set to 1.

Tag	id=2	id=3	id=4	id=5	id=8	id=11
conversionwarning	0	0	0	0	0	0
emph2	2	3	4	5	10	30
emph3	2	3	4	5	10	30
name	2	3	4	5	20	100
title	2	3	4	5	20	50
caption	2	3	4	5	10	10
collectionlink	2	3	4	5	10	10
languagelink	0	0	0	0	0	0
template	0	0	0	0	0	0

Table 5.1: Replication values used in the experiments.

5.5.1 Numerical results

The results are shown in table 5.2.
Key of the abbreviations used on the table:

- OR Gate (ML): or gate classifier, with the maximum likelihood approach.
- OR Gate (TI): or gate classifier, with the approach that considered interactions among terms.
- μBEP and μF1: micro averaged precision/recall breakeven point, micro averaged F1 measure, respectively.

Method	Reduction	Selection?	μBEP	MBEP	μF1	MF1
Naïve Bayes	Only text	no	0.76160	0.58608	0.78139	0.64324
Naïve Bayes	Only text	$\geq$ 2 docs.	0.72269	0.67379	0.77576	0.69309
Naïve Bayes	Only text	$\geq$ 3 docs.	0.69753	0.67467	0.76191	0.68856
Naïve Bayes	Adding_1	None	0.75829	0.56165	0.76668	0.58591
Naïve Bayes	Adding_1	$\geq$ 3 docs.	0.68505	0.66215	0.74650	0.65390
Naïve Bayes	Adding_2	None	0.73885	0.55134	0.74413	0.54971
Naïve Bayes	Adding_2	$\geq$ 3 docs.	0.66851	0.62747	0.71242	0.59286
Naïve Bayes	Adding_3	None	0.71756	0.53322	0.72571	0.51125
Naïve Bayes	Adding_3	$\geq$ 3 docs.	0.64985	0.59896	0.68079	0.53859
Naïve Bayes	Tagging_1	None	0.72745	0.49530	0.72999	0.50925
Naïve Bayes	Tagging_1	$\geq$ 3 docs.	0.65519	0.60254	0.71755	0.60594
Naïve Bayes	Repl. (id=2)	None	0.76005	0.64491	0.78233	0.66635
Naïve Bayes	Repl. (id=2)	$\geq$ 2 docs.	0.71270	0.68386	0.61321	0.73780
Naïve Bayes	Repl. (id=2)	$\geq$ 3 docs.	0.70916	0.68793	0.73270	0.65697
Naïve Bayes	Repl. (id=3)	None	0.75809	0.67327	0.77622	0.67101
Naïve Bayes	Repl. (id=4)	None	0.75921	0.69176	0.76968	0.67013
Naïve Bayes	Repl. (id=5)	None	0.75976	0.70045	0.76216	0.66412
Naïve Bayes	Repl. (id=8)	None	0.74406	0.69865	0.72728	0.61602
Naïve Bayes	Repl. (id=11)	None	0.72722	0.67965	0.71422	0.60451
OR Gate (ML)	Only text	None	0.37784	0.38222	0.59111	0.37818
OR Gate (TI)	Only text	None	0.79160	0.76946	0.79160	0.74922
OR Gate (TI)	Only text	$\geq$ 3 docs.	0.77916	0.78025	0.77916	0.73544
OR Gate (TI)	Only text	$\geq$ 2 docs.	**0.79253**	**0.78135**	**0.79253**	**0.75300**
OR Gate (ML)	Adding_1	None	0.40503	0.43058	0.58777	0.39361
OR Gate (ML)	Adding_1	$\geq$ 3 docs.	0.39141	0.41191	0.57809	0.36936
OR Gate (ML)	Adding_2	None	0.40573	0.43335	0.58908	0.39841
OR Gate (ML)	Adding_2	$\geq$ 3 docs.	0.39204	0.41490	0.57951	0.37346
OR Gate (ML)	Notext_2	None	0.40507	0.42914	0.48818	0.38736
OR Gate (ML)	Tagging_1	None	0.37859	0.40726	0.57274	0.35418
OR Gate (ML)	Tagging_1	$\geq$ 3 docs.	0.36871	0.38475	0.56030	0.32546
OR Gate (TI)	Tagging_1	None	0.73784	0.74066	0.73789	0.70121
OR Gate (TI)	Repl. (id=2)	None	0.78042	0.76158	0.78042	0.73768
OR Gate (TI)	Repl. (id=3)	None	0.78127	0.76095	0.78127	0.73756
OR Gate (TI)	Repl. (id=4)	None	0.78059	0.75971	0.78059	0.73511
OR Gate (TI)	Repl. (id=5)	None	0.77977	0.75833	0.77978	0.73245
OR Gate (TI)	Repl. (id=11)	None	0.77270	0.74943	0.77270	0.72186
OR Gate (TI)	Repl. (id=11)	None	0.73041	0.70260	0.73041	0.66733

Table 5.2: Results of our different approaches over the Wikipedia XML corpus.

- MBEP and MF1: macro averaged precision/recall breakeven point, macro averaged F1 measure, respectively.

- $\geq i$: only terms appearing in at least i documents are selected.

5.5.2 Conclusions from the Results

At a first sight, the best classifier in the four measures is a flat text classifier, the OR Gate in its TI version, without using any special document transformation (it seems than none of the proposed methods in section 5.4 help this classifier).

However, it is a clear fact that the "replication" approach helps the Naïve Bayes classifier. One of the main drawbacks of this classifier are the bad results obtained, generally in macro measures (due to the nature of the classifier, that benefits the classes with higher number of training examples). This fact can be solved easily using a replication approach as stated in the table of results.

On the other hand, adding and tagging methods do not seem to give good results, using these classifiers.

5.6 Final remarks

The main relevant results presented here are the following:

- We have shown different methods of representing structured documents into plain text ones. We must also recall that some of them are new.
- According to the results, we found that we could improve categorization of structured documents using a multinomial Naïve Bayes classifier, which is widely known and it is included in almost every text-mining software package, in combination with the replication method.

On the other hand, the present paper raises the following questions that can be stated as future lines of research:

- How are the results of our models compared with a SVM using only the text of the documents?
- Can the naive Bayes classifier be improved more using a more sophisticated feature selection method?
- Having in mind that the replication approach is the one that has given the best results, what are the optimum replication parameters that can be used in Wikipedia? In other words, what marks are more informative and how much?

- Is there a way to make a representation of the structure of documents that could be used to improve the results of the OR Gate classifier (specially in its more promising TI version)?

- Do the "adding", "tagging" and "no text" approaches help other categorization methods, like, for instance, Rocchio or SVMs?

Managing structure in this problem has been revealed as a difficult task. Besides, it is not really clear if the structure can make a good improvement of categorization results.

5.7 Linked Text Documents

The information present on the Internet is not a set of isolated and independent documents. Instead, a typical web page, apart from the textual content, includes in its content explicit links to other web pages. These links, called *hyperlinks* are shortcuts on the text to automatically navigate from the current document to the linked one. Although linked documents are not the only example of linked text documents, they are probably the most extensive and easy to understand one.

We can define a corpus of linked files as a corpus of flat text documents where a graph structure between the different documents is also given. Note that we have not specified the kind of the graph (directed or undirected). If the graph is made of explicit links, it is generally a directed graph (like for example, the case of hyperlinked documents). If the graph is inferred from underlying non-explicit relations it is normally an undirected graph (for example, two documents can be linked if they share the same author).

Thus, if we are given a graph of relationships among documents, along with a training (labeled) and a test corpus, we can observe the following facts:

1. Here, the classical **i.i.d. assumption on the corpus is violated**. If two documents are linked, they cannot be statistically independent.

2. In the more general setting, **the graph is only one**, linking training and test documents (i.e. you can have some information for some of the test

documents before accessing their content, inspecting only the links to or from labeled documents in the training set).

The first fact has a very direct conclusion: in order to make Document Categorization, we can extract some features of the linked documents (or the document that link one). A very common application of this principle is using the anchor text (the text associated to the link) as additional features of the linked document, adding some terms that may not occur on the document.

The second fact increases the dramatism of the previous affirmation. So, if we want to ignore all textual features, we could still make some categorizations of unlabeled documents, only looking at the labels of the neighbors, in order to infer the own label. From a certain point of view, we can see the labeling of the neigbors as features, and train a classifier on it. The methods that make use of these procedures (using or not textual features) are called, in general, *graph labeling* [25] procedures.

5.8 Link-based Categorization

5.8.1 Statement of the Problem

If we are given a set of training documents $\mathcal{D}$, where some graph structure (edges between pairs of documents) is also provided, the problem of Structured Document classification consists in finding one classifier f which, being based on the training data (documents and their relationships), will be able to categorize a new document D, using its textual information and the labeling of the graph (either the one formed with the labels of the training documents, or including the labels that are an output of our classifier in an intermediate step). That is to say, finding a classifier using the graph structure and the content of the documents, able to assign a new document D one or several categories $f(D)$ (depending on the kind of the problem chosen). Of course, the problem again can be binary, multiclass, or multilabel.

This means that we can look at the entire graph and its labeling in any moment of the categorization process. Obviously, this assumption is somewhat unrealistic for a real categorization task. If you think, for example, in a batch categorization

of web page documents, where a single page is given in every stage of the process, a more realistic assumption will be to consider only the information available in that precise moment (that is to say the set of documents linked by the one being categorized). Obtaining the inverse information (the set of documents that point to one) is not an easy task because it assumes we already know the entire graph on the beginning of the process (which will require a whole preprocessing of the corpus before doing categorization). Anyway, in some of the cases we shall only use output links, not always for this reason, but because they seem to be more informative.

Note that the problem of Link-based Document Categorization does not try to find the graph structure among the documents in any way. The set of graph nodes and edges is an input data, and our objective is only to find the labels of the nodes in the graph, but not its structure.

5.8.2 State of the Art

Contrary to the Structured Document Categorization, this is a very active field of research. The problem has been studied since 1998 in [27] where it was stated that classical categorization methods did not get good results for the corpora of linked documents.

The paper by Yang [137] gives a panoramic of Link-based Document Categorization, for the specific case of hyperlinked documents (web pages), presenting an extensive experimentation.

In the last years, there has been a notable movement in this community, in the subfield of *collective classification* [56]. This approach tries to solve the problem of graph labeling with an iterative method also using the content of the documents. First, some labels are estimated on the test set, and after that, a combination or propagation is done until some kind of convergence is obtained. Following the work presented in Getoor et al. [122], we can make a taxonomy of the several methods with some key references:

- Methods based on **Iterative Classification Algorithm** (ICA), which is a very simple approach, and uses the labeling of the neighbors to estimate the current label [23; 79; 80; 100].

- Methods using a **Relaxation Labelling algorithm**, a more advanced procedure, which makes use of the labeling of the neighbors, along with their probability to make more accurate estimates in each iteration [27; 81; 120; 121; 122].
- Methods using a **Gibbs Sampling procedure**, which try to find a good labeling, sampling each node label iteratively taking into account the probability distribution given by the label of the neighbors and the output of the classifier [82; 86; 121; 122].
- Methods using a **Loopy Belief Propagation procedure** (LBP), which use the Pearl's propagation algorithm and any of its variants [81; 120; 121; 122; 139].

For more details on these methods, we refer the reader to the reference [49].

5.8.3 Presentation of our Models

Our main direction in developing new models will be modeling with Bayesian networks the structure of the document to classify, and the labels of the neighbors. In our first model, which will be presented in section 5.9, the probabilities of the classes, given the document contents (the terms) are firstly computed using a base probabilistic classifier (for example, a Naïve Bayes). After that, we shall combine them with the labeling of the linked documents (either if they were from the training file, or from the test file and their labels are only estimated). The Bayesian network structure is, in this case, fixed, but the number of nodes is variable for each document (there will be a variable for each linked file). This is basically the model that we presented in [19].

Our second model, which will be presented in section 5.12, is rather more flexible, and it has been developed for the multilabel case. In this model, the structure of the Bayesian Network is not fixed, and it is learnt from data. This is a computationally harder approach but its results are, on the other hand, much better. For this case, the interactions represented are among the categories of the document itself, and the categories of the related (linked) documents (therefore, the number of variables will be fixed for each document, because we do not have

into consideration how many files of a certain category are linked). This model was presented in [20].

Recall that, in both cases, we use the notation "linked" to note that a document is related to any other. This does not necessarily means that the "linked" document is (hyper-)linked by the current document, and, in certain occasions, we could be ignoring the directions of the links, or using the incoming links, instead of the outgoing ones.

5.9 A New Model for Multiclass Link-based Classification Using Bayesian Networks

5.9.1 The Basic Model

We present here a new model [19] developed to solve this problem in its multiclass version [38]. The method is an extension of a probabilistic classifier (we shall use in the experiments the Naïve Bayes classifier (in its multiclass version, i.e. the variable C ranges in $\{c_0, c_1, \ldots, c_n\}$, but other probabilistic classifiers could also be employed) where the evidence is not only the document to classify, but this document together with the set of related documents. Note that, in principle, we shall try to use only information which is available in a natural way for a text classifier. Considering that different documents are processed through batch processing, the information easily available to a system, given a document, is the set of documents it links (not the set of documents that link it).

Consider a document d_0, which is linked to documents $d_1, \ldots, d_m$. We shall consider the random variables $C_0, C_1, \ldots, C_m$, all of them taking values in the set of possible category labels $\{c_0, c_1, \ldots, c_n\}$. Each variable C_i represents the event "The class of document d_i is". Let e_i be the evidence available concerning the possible classification of each document d_i (the set of terms used to index the document d_i or the class label of d_i). The proposed model can be graphically represented as the Bayesian network displayed in figure 5.9.

The independencies represented by the Bayesian network are the following: given the true class of the document we want to classify, the categories of the linked documents are independent among each other. Moreover, given the true

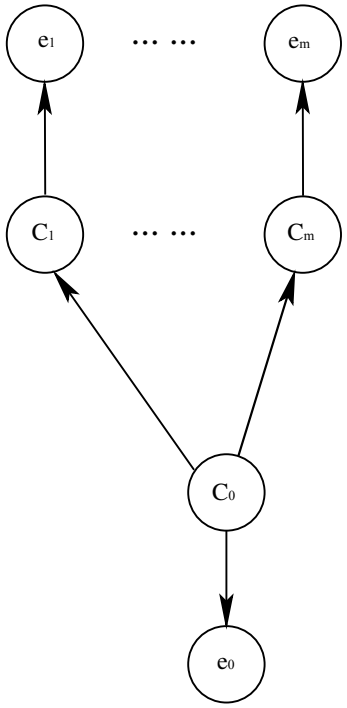

Figure 5.9: Bayesian network representing the proposed model.

category of a linked document, the evidence about this category due to the document content is independent of the original category of the document we want to classify.

Our objective is to compute the posterior probability $p(C_0|e)$, where e is all the available evidence concerning document d_0, $e = \{e_0, e_1, \ldots, e_m\}$. We can start computing this value using Bayes' rule:

$$p(C_0 = c_0|e) = \frac{p(C_0 = c_0)\, p(e|C_0 = c_0)}{p(e)} \propto p(C_0 = c_0)\, p(e_0, e_1, \ldots, e_m|C_0 = c_0). \tag{5.1}$$

As it can be seen on figure 5.9, the variable C_0 separates all the different e_i making them independent, so the joint distribution becomes:

$$p(e_0, e_1, \ldots, e_m|C_0 = c_0) = \prod_{i=0}^{m} p(e_i|C_0 = c_0),$$

And we can rewrite equation 5.1 as follows:

$$p(C_0 = c_0|e) \propto p(C_0 = c_0) \prod_{i=0}^{m} p(e_i|C_0 = c_0) \propto p(C_0 = c_0|e_0) \prod_{i=1}^{m} p(e_i|C_0 = c_0). \tag{5.2}$$

Each individual value $p(e_i|C_0 = c_0)$ can be expressed, using the law of total probability, and taking as exclusive events $C_i = c_0, C_i = c_1, \ldots, C_i = c_n$ (with $i > 0$) as this:

$$p(e_i|C_0 = c_0) = \sum_{c_j = \{c_0, c_1, \ldots, c_n\}} p(e_i|C_i = c_j, C_0 = c_0)\, p(C_i = c_j|C_0 = c_0) \tag{5.3}$$

Recall that variables C_i (with $i > 0$) separate e_i and C_0, making $p(e_i|C_i = c_j, C_0 = c_0) = p(e_i|C_i = c_j)$. Applying Bayes' rule, we can write the equation 5.3 as:

$$p(e_i|C_0 = c_0) = p(e_i) \sum_{c_j = \{c_0, c_1, \ldots, c_n\}} \frac{p(C_i = c_j|e_i)}{p(C_i = c_j)}\, p(C_i = c_j|C_0 = c_0) \tag{5.4}$$

From equations 5.4 and 5.2, we can give a final expression (equation 5.5) for $p(C_0 = c_0|e)$ using only values that can be available.

$$p(C_0 = c_0|e) \propto p(C_0 = c_0|e_0) \prod_{i=1}^{m} \left(\sum_{c_j = \{c_0, c_1, \ldots, c_n\}} p(C_i = c_j|C_0 = c_0)\, \frac{p(C_i = c_j|e_i)}{p(C_i = c_j)} \right) \tag{5.5}$$

As we can observe in equation 5.5, the posterior probability of C_0 has two components: a part which only depends on the evidence associated to the document d_0 to be classified ($p(C_0 = c_0|e_0)$) and another part related with the information about the class labels of each one of the documents linked with d_0, which can be obtained using its own local evidence ($p(C_i = c_i|e_i)$). This information is combined with the estimated probabilities of a linked document being of class c_i

given that the document linking to it is of class c_0.

The posterior probabilities $p(C_0 = c_0|e_0)$ and $p(C_i = c_i|e_i)$ can be obtained using some standard probabilistic classifier, whereas the probabilities $p(C_i = c_i)$ and $p(C_i = c_i|C_0 = c_0)$ can be estimated from the training data simply by following these formulas:

$$p(C_i = c_i) = \frac{N_i}{N}$$

and

$$p(C_i = c_i|C_0 = c_0) = \frac{L_{0i} + 1}{L_{0\bullet} + |C|}$$

where N_i is the number of training documents classified by category i, N is the total number of documents, L_{0i} is the number of links from documents of category 0 to category i, $L_{0\bullet}$ is the total number of links from documents of category 0, and $|C|$ is the number of categories. Note that in the estimation of $p(C_i = c_i|C_0 = c_0)$ we have used Laplace smoothing. In all our posterior experiments, using Laplace gives better results than not using it.

Therefore, we can think of the proposed model as a method to modify the results offered by a base probabilistic classifier taking into account the information available about the linked documents and the relationships between categories (the prior probabilities $p(C_i = c_i)$ and the values $p(C_i = c_i|C_0 = c_0)$).

5.9.2 Extension to Inlinks and Undirected Links

The independences represented by the Bayesian network are not directly related to the direction of the links. Instead of outlinks, we could think of the previous model as a model that takes into consideration the incoming links. Thus, the C_i and e_i ($i > 0$) variables would represent the documents that link to one (instead of the files linked by one), and the formula (5.5) would still be valid. In the case of the incoming links, we should reestimate the dependencies among categories as follows:

$$p(C_i = c_i|C_0 = c_0) = \frac{L_{i0} + 1}{L_{\bullet 0} + |C|}$$

where L_{i0} is, as previously stated, the number of links from documents of category i to category 0, and $L_{\bullet 0}$ is the total number of links to documents of category 0.

Moreover, in the collective classification literature, the direction of the links is often not considered, so, we also could propose a model where C_i and e_i $(i > 0)$ represent the documents linked or being linked (that is to say, neighboured) by the document to classify. In that case, the probabilities would be these:

$$p(C_i = c_i | C_0 = c_0) = \frac{L_{i0} + L_{0i} + 1}{L_{\bullet 0} + L_{0\bullet} + |C|}.$$

Therefore, these would be our three models: the original one (with incoming links), and the extensions using outlinks and undirected links.

5.10 Experiments on the INEX'08 Corpus

5.10.1 Study of the Corpus

The INEX'08 Document Mining corpus [38] is a document collection for what it was developed our model presented in 5.9. It is composed of 114336 documents extracted from the Wikipedia XML Corpus [37], with a size of the graph (number of edges) of 636187. The percentage of training documents is more or less 10% of the documents (there is 11437 training documents), with the 90% for testing purposes. The number of categories of this corpus is 15. This corpus is also a single label one (that is to say, we are dealing with a multiclass categorization problem, where each document is assigned to only one category).

As we are testing a model which makes use only of the link information, we shall ignore the structural information, represented by the XML markup of the files. This information could also be used to improve classification with a model like that described in section 5.4, but in this case we shall not use it. In fact, we shall see that information from the links can be a very valuable data.

Before making any experiment with categorization models, we shall try to have a deeper look at the information provided by the graph of links. The fundamental question is whether those links are supposed to help in the final process of text categorization. Given the nature of the corpus (documents coming from an

Encyclopedia), we could think that articles on one category should tend to link articles on the same category. This kind of dependence can clearly be represented by our model, as we shall see later.

A careful review of the different kinds of dependencies represented by hyperlinks (*regularities*) is given by Yang [137], and following her terminology we can state that we should be in an "encyclopedia regularity". We reproduce here her definition:

> *One of the simplest regularities is that certain documents with a class label only link with documents with the same class label. This regularity can be approximately found in encyclopedia corpus, since encyclopedia articles generally reference other articles which are topically similar.*

In order to verify experimentally this fact, we have plotted, in figure 5.10, a matrix where the rows and columns are one of the 15 categories. Each matrix value $m_{i,j}$ represents the probability that a document of class i links a document of class j, estimated from the training document collection.

As it can be seen (the matrix has a strong weight in its diagonal), documents of one category tend to link documents of the same category. Moreover, doing the same plot with the probability that a document of class i being linked by a document of class j, and another one with the probability of a document of class i links or is linked by a document of class j, we obtain a similar result (a matrix with a high weight for the diagonal values).

Thus, although we could think that only the outlinks tend to be useful, we can affirm that also inlinks are useful, and also consider the links without any direction.

5.10.2 Experimental Results

With this experiment, we are going to study if our model can be above a certain baseline (a flat classifier). First of all we study the two executions on the test set, a baseline (Multinomial Naïve Bayes) and our multiclass method (combined with Naïve Bayes). The results obtained are the following (the evaluation measure used is the recall, measuring the proportion of correct assigned labels):

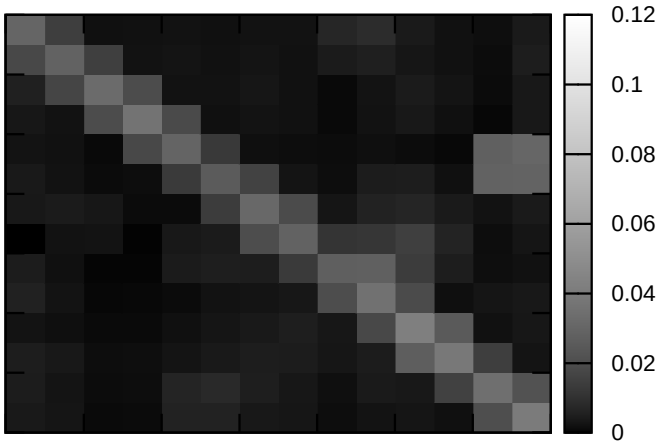

Figure 5.10: Probability that a document of class i links a document of class j, on the INEX'08 corpus.

- A classical *Naïve Bayes algorithm* on the flat text documents obtained 0.67674 of recall.
- *Our proposal* using the previous Naïve Bayes as the base classifier obtained 0.6787 of recall (using outlinks).
- Our model (inlinks): 0.67894 of recall.
- Our model (neighbours): 0.68273 of recall.

Although all our methods improve the baseline, the results achieved are not really significant. In order to justify the value of our model, we are asking now ourselves which is the predicting power of our proposal, by making some additional computations in an "ideal setting". This "ideal setting" is, for a document being classified, to be surrounded (linking, linked by or both of them) with documents whose class membership is perfectly known (and hence we can set for a related document d_k of category c_i, $P(C_k = c_i|d_k) = 1$ -the true class- and $P(C_k = c_j|d_k) = 0$ -the false categories- $\forall c_j \neq c_i$). Remember that, in previous

experiments, a surrounding document whose category was not known should be first classified by Naïve Bayes, and then that estimation (the output probability values) was used in our model.

So, the procedure is the following: for each document to classify, look at the surrounding files. For each one, if it is a training file, use that information (perfect knowledge), and if it is a test file, use also its categorization information taken from the test set labels to have our file related to documents with perfect knowledge. This "acquired" knowledge is obviously removed for the next document classification.

In this "ideal setting" we have made two experiments: one combining naïve Bayes with our model (like the second one of the previous two), and one which combined a "blind classifier" (the one that gives equal probability to each category) with our model. The first should be better than the two previous ones, and the second one could give us an idea of the true contribution to the predictive power of our model, despite the underlying basic classifier used.

- Model for outlinks in an "ideal setting" using Naïve Bayes as a base classifier: 0.69553 of recall.
- Model for outlinks in an "ideal setting" using a "blind classifier": 0.46500 of recall.
- Model for inlinks in an "ideal setting" using Naïve Bayes as a base classifier: 0.69362 of recall.
- Model for inlinks in an "ideal setting" using a "blind classifier": 0.73278 of recall.
- Model for neighbours in an "ideal setting" using Naïve Bayes as a base classifier: 0.70212 of recall.
- Model for neighbours in an "ideal setting" using a "blind classifier": 0.66271 of recall.

The first experiment provides the desired result: the recall is improved (although not so much). The small improvement could be due, in some part, to the

extreme values given in this corpus by the Naïve Bayes classifier (very close to 0 and 1). The introduction of these values in equation 5.5, as the first factor in the final posterior probability of each document, makes difficult to take into account (in the categories of the values close to 0) the information provided by the second factor (the combination of the information given by all the linked files), vanishing in some cases because of the low value of the first factor.

However, the second experiment showed us that, only using link information, and ignoring all content information of the document to classify, in this "ideal setting" of knowing the true class of each surrounding document, our method can reach 0.46500, 0.73278 or 0.66271 of recall. In the case of the inlinks, ignoring the content of the document to classify and perfectly knowing the values of the categories of the surrounding documents, gives better results than using this content. Besides, these values are clearly high, which gives us the idea of the predictive power of link information in this problem.

5.11 Conclusions and Future Works

We have proposed a new model for classification of multiclass linked documents, based on Bayesian networks. We have also justified the possibly good performance of the model in an "ideal" environment, with some promising results.

To improve those results in the future, we could use a classifier (probabilistic) with a better performance. Such a classifier could be a logistic regression procedure, a higher dependence network or just a SVM with probabilistic output (using Platt's algorithm [103]). The probability assignments should also be "softer", in the sense that several categories should receive positive probability (naïve Bayes tended to concentrate all the probability in one category, zeroing the others and making the information provided by the links not useful, in some way).

As future work we would like to study this problem as a collective classification problem (see section 5.8.2), and try to apply this method in one of the particular solutions (those that need a "local classifier") that are being given to it.

5.12 A New Model for Multilabel Link-based Classification Using Bayesian Networks

This section describes a new methodology that models a link-based categorization environment using Bayesian networks, for the problem of multilabel categorization. This problem is different and more complex than the previus one, and therefore we shall develop a new model appropriate for it.

In this development, we shall only use data from incoming links, because we carried several experiments on the corpus, and found them much more informative than outgoing ones. Anyway, information from outgoing links (or even considering undirected links) could also be used in this model.

5.12.1 Modeling link structure between documents

In this problem, we shall deal with a multilabel corpus. The methodology proposed by the state-of-the art is to treat this problem as several independent binary classifiers. In this case, the classifiers should not really be independent, because we assume that, in a general setting, there can be explicit relationships among different categories which makes this problem even more difficult. In fact, as we will see in 5.13, the corpus includes documents coming from different categories which form a hierarchy, and relationships (links) among documents of different categories are surely present on the collection.

We shall build automatically from data, a Bayesian network-based model which could represent almost[1] all the relationships among the categories of a certain document, and the set of categories present on the related (linked) documents.

Therefore, for a problem of n categories, we will consider the following set of $2n$ random variables: first of all, for every category i we will have one binary variable C_i, representing that the current document belongs or not to category

[1] A Bayesian network cannot represent some independences in a set of random variables, and the learning method that we will choose will search in a restricted search space, limiting the number of parents of each node. Anyway, this model could possibly represent the majority of the relationships present on the data.

C_i. That is to say, variable C_i ranges in states $\{c_i, \overline{c}_i\}$, and models the probability of a document being (or not) of class C_i.

On the other hand, each category will have another associated random variable: LC_i, (with states $\{lc_i, \overline{lc}_i\}$), representing if there is a link, or not, from documents of category i to the current document. This set of variables is to sum up the labeling of the neighbors, but in this case, its size does not depend on the number of surrounding documents of the current document to classify (it is fixed).

Therefore, to learn a model from the data, we shall use the training documents, each one as an instance whose categories (values for variables C_i) are perfectly known, and the links from other documents. If a document is linked by another training document of category j, we shall set $LC_j = lc_j$, setting it to $\overline{lc}_j$ otherwise. Note that a training document could be linked by test documents (whose categories are unknown). In that case, this evidence is ignored, and categories which do not have any document linked to the current document, are set their variables to $\overline{lc}_j$.

So, we could learn a Bayesian network from training data (see next subsection) and, for each test document d_j, we could compute $p(c_i|e_j)$, where e_j represents all the evidence given by the information of documents that link this.

Thus, the question is the following: for a certain document d_j, given $p(c_i|d_j)$ and $p(c_i|e_j)$, how could we combine them in an easy way? We want to compute the posterior probability $p(c_i|d_j, e_j)$, the probability of a category given the terms composing the document and the evidence due to link information.

Using Bayes' rule, and assuming that the content and the link information are independent given the category, we get:

$$\begin{aligned} p(c_i|d_j, e_j) &= \frac{p(d_j, e_j|c_i)\, p(c_i)}{p(d_j, e_j)} = \frac{p(d_j|c_i)\, p(e_j|c_i)\, p(c_i)}{p(d_j, e_j)} \\ &= \frac{p(c_i|d_j)\, p(d_j)\, p(e_j|c_i)\, p(c_i)}{p(c_i)\, p(d_j, e_j)} = \frac{p(c_i|d_j)\, p(d_j)\, p(c_i|e_j)\, p(e_j)}{p(c_i)\, p(d_j, e_j)} \\ &= \left(\frac{p(d_j)\, p(e_j)}{p(d_j, e_j)}\right)\left(\frac{p(c_i|d_j)\, p(c_i|e_j)}{p(c_i)}\right). \end{aligned}$$

The first term of the product is a factor which does not depend on the category. So, we can write the probability as:

$$p(c_i|d_j, e_j) \propto \frac{p(c_i|d_j)\, p(c_i|e_j)}{p(c_i)}$$

And we can rewrite the first expression in this final form:

$$p(c_i|d_j, e_j) = \frac{p(c_i|d_j)\, p(c_i|e_j) \,/\, p(c_i)}{p(c_i|d_j)\, p(c_i|e_j) \,/\, p(c_i) \;+\; p(\overline{c}_i|d_j)\, p(\overline{c}_i|e_j) \,/\, p(\overline{c}_i)} \tag{5.6}$$

We must make some final comments about this equation to make it more clear:

- As we said before, the posterior probability $p(c_i|d_j)$ is the one obtained from a binary probabilistic classifier, which is going to be combined with the information obtained from the link evidence. In the experiments section 5.13 we shall be specific about what probabilistic classifier we shall use.
- The prior probability used here, $p(c_i)$, is the one computed with propagation over the Bayesian network learnt with link information.
- Because the variables C_i are binary, it is clear that $p(\overline{c}_i|e_j) = 1 - p(c_i|e_j)$, $p(\overline{c}_i) = 1 - p(c_i)$ and $p(\overline{c}_i|d_j) = 1 - p(c_i|d_j)$.

5.12.2 Learning the link structure

Given the previous variable setting, from the training documents, their labels and the link file, we can obtain a training set for the Bayesian network learning problem, composed of vectors of binary variables C_i and LC_i (one for each training document).

We have used WEKA package [131] to learn a generic Bayesian network (not a classifier) using a hill climbing algorithm (with the classical operators of addition, deletion and reversal of arcs) [11], with the BDeu metric [53]. In order to reduce the search space, we have limited the number of parents of each node to a maximum of 3.

Once the network has been learnt, we have converted it to the Elvira [47] format. Elvira is a software developed by some Spanish researchers which has implemented many algorithms for Bayesian networks. In this case, we have used it to carry out the inference procedure. This is done as follows:

1. For each test document d_j, we set in the Bayesian network the LC_i variables to lc_i or $\overline{lc_i}$, depending whether d_j is linked by at least one document of category i, or not, respectively. This is the evidence coming from the links (represented before by e_j).

2. For each category variable, C_i, we compute the posterior probability $p(c_i|e_j)$. This procedure is what is called *evidence propagation*.

Due to the size of the problem, instead of exact inference, we have used an approximate inference algorithm [13], firstly to compute prior probabilities of each category in the network, $p(c_i)$, and secondly, to compute the probabilities of each category given the link evidence e_j, for each document d_j in the test set, $p(c_i|e_j)$. The algorithm used is called Importance Sampling algorithm, and is faster than other exact approaches.

5.13 Experiments on the INEX'09 Corpus

5.13.1 Study of the Corpus

The INEX'09 XML Document mining corpus is a multilabel corpus, with a different nature than the INEX'08 one. While the INEX'08 one came from a 2005 Wikipedia snapshot [39] converted to XML, the 2009 corpus comes from a more recent snapshot where the documents tend to be longer, and the set of links is surely more dense. The XML markup is different, but we shall ignore it as we did in 5.9, because we are only trying to figure out how link information help to improve categorization results.

For this corpus there are 54572 documents, corresponding to a test/train split of 10968 documents in the training corpus (about 20% of the total), and 43604 in the test one. The number of categories is 39, and we are given their names.

Also, a link file is shown, which gives specific relations among documents (which may be in the training corpus or not).

Therefore, the main difference between this INEX track in 2008 and 2009 is the fact that the training corpus is made of multilabeled documents, that is to say, a document can belong to one or more categories. The rest of the rules are esentially the same, although the document collection and the set of categories are also different.

Several experiments in the same direction that those done in 5.10 showed us the same fact for the 2009 corpus, although we do not reproduce them here. Apart from those experiments, the names of the categories (which are explicitly given in the training set), tend to show categories which are probably coming from a hierarchy (for example `Portal:Religion`, `Portal:Christianity` and `Portal:Catholicism`). The two known facts about the relations are summarized here:

- In this linked corpus, due to its nature, a "hyperlink regularity" is supposed to arise [137].
- There are some categories strongly related a priori, because the probable existence of a (partially unknown) hierarchy.

5.13.2 Results

To make the experiments, we have used, as base classifier two different probabilistic classifiers. The Bayesian network classifier will use their values as a base to compute the final probability, as stated in equation 5.6.

First of all, the binary Naïve Bayes, in its multinomial version (see chapter 1 for more details), and the "term interaction" version of our OR gate classifier (explained in chapter 3). They will be also used as baseline, and will be noted as "NB" (for the Naïve Bayes) and "OR" (for the OR gate).

The performance measures were suggested by the corpus authors [37], and they are Accuracy (ACC), Area under Roc curve (ROC), F1 measure (PRF) and Mean average precision by document (MAP). As stated in the first chapter, some of them will measure "hard categorization", and some "soft categorization".

5.13.2.1 Results without scaling

The results of the models on the testing set are displayed in table 5.3, where M means the "macro" version of the measure, and μ means the "micro" one.

	MACC	**μACC**	**MROC**	**μROC**	**MPRF**	**μPRF**	**MAP**
NB	0.95142	0.93284	0.80260	0.81992	0.49613	0.52670	0.64097
NB + BN	0.95235	0.93386	0.80209	0.81974	0.50015	0.53029	0.64235
OR	0.75420	0.67806	0.92526	0.92163	0.25310	0.26268	0.72955
OR + BN	0.84768	0.81891	0.92810	0.92739	0.31611	0.36036	0.72508

Table 5.3: Preliminary results.

In both cases, the Bayesian network version of the classifier outperforms the "flat" version, though the results of the OR gate are surprisingly poor in ACC and PRF. This fact is due to the nature of the classifier, and to the kind of evaluation: For the OR gate, the fact that $p(c_i|d_j, e_j) > 0.5$ holds does not mean necessarily that d_j should be labeled C_i, whereas in the Naïve Bayes does (this was the criterion used by the evaluation procedure to assign categories to the test documents).

In fact, for the OR gate classifier is not known, a priori, what is the appropriate threshold τ_i that assigns d_j to class C_i if $p(c_i|d_j, e_j) > \tau_i$. This is not a major problem to compute, for example, averaged break-even point measures [119], where no hard categorization is needed. In this case, the threshold 0.5 has been adopted, and we need to re-adapt the model to this setting in order to perform well.

In the following section we can see how we estimated a set of thresholds (using only training data) and how we scaled the probability values, in order to match the evaluation criteria, dramatically improving the results.

5.13.2.2 Scaled version of the Bayesian OR gate results

To make this version of the OR gate results, we have followed this procedure: using only training data, a classifier has been built (both in its flat and Bayesian network versions), and evaluated using cross validation (with five folds). In each fold, for each category, we have searched for the threshold of probability that

gives the higher $F1$ measure per class and, afterwards, all thresholds have been averaged over the set of cross validation folds.

This is what is called in the literature the *Scut* thresholding strategy [136]. Thus, we obtain, for each category a threshold τ_i between 0 and 1 (different for each of the two models). We should then to transform the results to a scale where each category threshold is mapped to 0.5.

So, the probabilities of the or gates are rescaled using a linear continuous function f_i, which verifies $f_i(0) = 0$, $f_i(1) = 1$ and $f_i(\tau_i) = 0.5$. The function is:

$$f_i(x) = \begin{cases} \frac{0.5x}{\tau_i} & \text{if } x \in [0, \tau_i] \\ 1 - \frac{0.5}{1-\tau_i}(1-x) & \text{if } x \in (\tau_i, 1] \end{cases}$$

Then, the new probability values are computed, using the old values $p(c_i|d_j, e_j)$, as $\hat{p}(c_i|d_j, e_j) = f_i(p(c_i|d_j, e_j))$. Once again, we would like to recall that these new results are only "scaled" versions of the old ones, with thresholds being computed only using the training set. The new results are displayed in table 5.4.

	MACC	**μACC**	**MROC**	**μROC**	**MPRF**	**μPRF**	**MAP**
OR	0.92932	0.92612	0.92526	0.92163	0.45966	0.50407	0.72955
OR + BN	0.96607	0.95588	0.92810	0.92739	0.51729	0.55116	0.72508

Table 5.4: Results using thresholds.

Note that, using the scaling procedure, ROC and MAP values remains equal, whereas PRF and ACC, on the contrary, are improved considerably, and gives results which are much more better.

5.14 Conclusions and future works

Although in section 5.9 we proposed a method that captures some "fixed" relations among categories, given this different problem setting and its higher dimensionality, for the multilabel we have learnt those relations automatically from data, leading to a more flexible approach.

Given the previous results, we can state the following conclusion: the use of the Bayesian network structure for links can improve noticeably a basic "flat-text" classifier, for the case of a multilabel corpus.

This statement is clear, particularly on the case of the OR gate classifier, where some measures, like micro PRF are improved near 10%. Accuracy is improved 3-4%, while ROC stands more or less equal. Only MAP is slightly decreased (less than 1%). The changes on the Naïve Bayes classifier are more irrelevant, but they are all positive too.

The results could probably be improved with the usage of a better probabilistic base classifier. For example, a logistic regression or some probabilistic version of a SVM classifier (like the one proposed by Platt [103]), which are likely to have better results than our base models (although they can be much more inefficient). We expect to carry out more experiments with different basic classifiers on this corpus in the future.

Part III

Conclusions

Conclusions and Future Works

This last chapter presents the general conclusions of the dissertation. We recall that the specific conclusions of each contribution were previously given at the end of each corresponding chapter. Besides, two lists of publications are included (of the results presented in this thesis, and other publications not directly related with this content), together with some future works.

In chapter 3, we presented the OR gate-based classifier, as an alternative to the multinomial version of the Naïve Bayes classifier. We think it is a lightweight approach to classification, fast for both training and classification purposes, and easily updatable (in the case that new training information were made available, once trained the classifier). Moreover, the pruning procedure explained there gives, with the supporting experimentation, some evidence that our classifier is reasonably better than the Naïve Bayes approach (and other classic classifiers). In particular, macro measures obtain really promising results, indicating that no category is left apart, even those being very few populated (which is precisely the Achilles' heel of the Naïve Bayes).

The developments on the thesaurus-based categorization, shown in chapter 4 are undoubtly very clear: on the one hand, we can firmly state that the information present on the thesaurus (metadata and relationships) is very relevant in order to make categorization (and it is relatively well captured by our model), and on the other hand, that classic models fail when treating this problem as a plain supervised categorization one, because of the huge number of categories and the lack of training examples. The experiments presented with the parliamentary initiatives collection are very extensive, and show that the Naïve Bayes or even the Linear SVM (the state-of-the-art in Document Categorization) are clearly outperformed by our Bayesian network-based model. Furthermore, the chapter

has presented the problem of thesaurus-based automatic indexing, which is –in our opinion– an achievement by itself, and opens a new subfield of research in Document Categorization.

The Structured Document Categorization results, presented in chapter 5, are of diverse consideration. First of all, our approach to the XML categorization problem seems to be useful, but model-dependent (the OR gate classifier does not benefit from the "text replication" approach despite the Naïve Bayes does). For the problem of link-based categorization, we can state that the main lesson learnt is that the labels of the neighboring documents help to categorize the current one. In that problem, we have tried a fixed-structure Bayesian network approach, and a learnt-structure one, being the latter the most promising one (though the former also improved a baseline). In the link-based categorization, our method is a new approximation to this problem, and again, with a different method (Bayesian networks, in our case).

List of Publications

The different studies included in this dissertation have been presented in the following publications:

1. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero: *Automatic Indexing from a Thesaurus Using Bayesian Networks: Application to the Classification of Parliamentary Initiatives*. In: Khaled Mellouli (Ed.): Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 9th European Conference, ECSQARU 2007, Hammamet, Tunisia, October 31 - November 2, 2007, Proceedings. Lecture Notes in Computer Science **4724**, pp. 865–877, Springer 2007, ISBN 978-3-540-75255-4.

2. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Probabilistic Methods for Structured Document Classification at INEX'07*. In: Norbert Fuhr, Jaap Kamps, Mounia Lalmas, Andrew Trotman (Eds.): Focused Access to XML Documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle,

Germany, December 17-19, 2007. Selected Papers. Lecture Notes in Computer Science **4862**, pp. 195–206, Springer 2008, ISBN 978-3-540-85901-7.

3. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Or gate Bayesian networks for text classification: A discriminative alternative approach to multinomial naive Bayes*, actas del XIV Congreso Español sobre Tecnologías y Lógica Fuzzy, ESTYLF 2008, Mieres-Langreo (España), 17–19 Septiembre 2008.

4. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Thesaurus-based Automatic Indexing*. In: Handbook of Research on Text and Web Mining Technologies. Chapter XX, Vol. I, M. Song, Y.F. Brook Wu (Eds.), pp. 331–345, 2009. IGI Global. (ISBN 978-1-59904-990-8).

5. L. M. de Campos, A. E. Romero, *Bayesian network models for hierarchical text classification from a thesaurus*. Int. J. Approx. Reasoning **50**(7): 932-944 (2009).

6. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Probabilistic Methods for Link-Based Classification at INEX 2008*. In: Shlomo Geva, Jaap Kamps, Andrew Trotman (Eds.): Advances in Focused Retrieval, 7th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2008, Dagstuhl Castle, Germany, December 15-18, 2008. Revised and Selected Papers. Lecture Notes in Computer Science **5631**, pp. 453–459, Springer 2009, ISBN 978-3-642-03760-3.

7. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. R. Masegosa, A. E. Romero, *Link-based document classification using Bayesian Networks*. Proceedings of INEX 2009 (to appear), 2009.

The following references are publications of the author dealing with subjects that are not included in this dissertation:

8. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Un Prototipo de Motor de Búsqueda para los Diarios de Sesiones del Parlamento de Andalucía basado en Modelos Gráficos Probabilísticos*, Actas

de las VI Jornadas de Transferencia Tecnológica de Inteligencia Artificial (TTIA'2005), 37-44, ISBN: 84-9732-435-8, 2005.

9. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Garnata: An Information Retrieval System for Structured Documents based on Probabilistic Graphical Models*, Proc. of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2006), 1024–1031, Paris (France).

10. A. E. Romero, *Geometry and Information Retrieval*, International Seminar on Applied Geometry in Andalusia, Granada 2006 (Satellite of the ICM, Madrid 2006).

11. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Garnata Implementing the SID and CID Models at INEX'06*. In: Norbert Fuhr, Mounia Lalmas, Andrew Trotman (Eds.): Comparative Evaluation of XML Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, December 17-20, 2006, Revised and Selected Papers. Lecture Notes in Computer Science **4518**, pp. 165–177, Springer 2007, ISBN 978-3-540-73887-9.

12. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *A flexible object-oriented system for teaching and learning structured IR*. Proc. of the First International Workshop on Teaching and Learning of Information Retrieval (TLIR 2007), pp. 32–38, 2007.

13. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, C. Martín, A. E. Romero, *An Information Retrieval System for Parliamentary XML Documents based on Probabilistic Graphical Models*. In: O. Pourret, P. Naim, B. Marcot (Eds.), Bayesian Networks: A practical guide to applications, pp. 203–223, John Wiley & sons, ISBN: 978-0-470-06030-8, 2008.

14. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, C. Martín, A. E. Romero, *The Garnata Information Retrieval System at INEX'07*. In: Norbert Fuhr, Jaap Kamps, Mounia Lalmas, Andrew Trotman (Eds.): Focused

Access to XML Documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle, Germany, December 17-19, 2007. Selected Papers. Lecture Notes in Computer Science **4862**, pp. 57–69, Springer 2008, ISBN 978-3-540-85901-7.

15. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, C. J. Martín-Dancausa, A. E. Romero, *New Utility Models for the Garnata Information Retrieval System at INEX'08*. In: Shlomo Geva, Jaap Kamps, Andrew Trotman (Eds.): Advances in Focused Retrieval, 7th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2008, Dagstuhl Castle, Germany, December 15-18, 2008. Revised and Selected Papers. Lecture Notes in Computer Science **5631**, pp. 39–45, Springer 2009, ISBN 978-3-642-03760-3.

Future Work

We present, here in this section, some ideas to continue developing this work in the future. In first place, concerning the OR gate-based classifier, we think that several different approaches in weight training could be tested. In fact, we have seen that the methods that operate a conditional likelihood minimization [63; 129], though computationally hard, work reasonably well, even reaching the levels of the Linear SVM. Perhaps, a hybrid methodology for the weight training could keep a lightweight procedure in the training stage (i.e. not introducing complexity), and could improve the results on the classification part. Indeed, the question that arises is that wether we could approach to the categorization power of the Linear SVM algorithm, which is the state-of-the-art in Document Categorization, giving outstanding results in all the benchmark collections. In order to extend our model, we also think that other canonical models [42] could be used in this task (as, for example the noisy MAX gate, the noisy AND, or a combination among different canonical models).

The work in thesaurus-based categorization could be extended to deal with the particularities of very specific thesaurus. For the case of the medical thesaurus, MeSH [55], our model could be improved with only doing very few changes, in

order to represent some very small differences that this thesaurus has, compared with others. Moreover, a more extensive experimentation could be done with the presented models, in a larger corpus (as, for example. the Acquis corpus, categorized with the Eurovoc thesaurus). With this experimentation, a training model for the different free parameters (weights), could also be developed, in order to improve our categorization results, and make a categorization algorithm free of any manually-specified values.

Finally, the set of works developed with structured categorization could be extended in several ways. For the XML categorization, we think that finding one way to obtain the optimal replication values (in the "text replication" approach) could be interesting, in order to make more solid conclusions in this field. For the link-based categorization problem, there is a room for improvement, in the approach where the Bayesian network was learnt, because some other learning and propagation algorithms could be tested, combined with a different and richer neighborhood representation of the document (instead of just binary vectors).

Conclusiones y Trabajos Futuros

Este último capítulo presenta las conclusiones generales de la memoria. Debe tenerse en cuenta que las conclusiones específicas de cada contribución ya se dieron anteriormente al final del capítulo correspondiente. También se incluyen dos listas de publicaciones (de los resultados presentados en esta tesis y de otras publicaciones no tratadas aquí), junto con un listado de posibles trabajos futuros.

En el capítulo 3, presentamos el clasificador basado en puertas OR como una alternativa a la versión multinomial del clasificador Naïve Bayes. Creemos que es un enfoque para clasificación, computacionalmente hablando, muy ligero, tanto en el aprendizaje, como en el proceso de clasificación propiamente dicho, y también fácilmente actualizable (para el caso en el que nueva información de entrenamiento aparezca una vez entrenado el clasificador). Además, el procedimiento de poda explicado ahí da, junto a la experimentación que lo justifica, cierta evidencia de que nuestro clasificador es razonablemente mejor que el enfoque Naïve Bayes (y otros clasificadores clásicos). En particular, en las medidas "macro", obtenemos resultados muy prometedores, indicando que no se perjudica demasiado ninguna categoría, incluso a aquellas que están poco pobladas (que es precisamente el talón de Aquiles del Naïve Bayes).

Los avances realizados en clasificación basada en tesauros, mostrados en el capítulo 4 son ciertamente muy claros: por una parte podemos constatar que la información presente en un tesauro (metadatos y relaciones) es muy relevante para hacer clasificación (y se captura relativamente bien por nuestro modelo), y por otra parte, que los modelos clásicos fallan al tratar este problema como un problema de clasificación supervisada normal debido al alto número de categorías y a la falta de ejemplos de entrenamiento. Los experimentos realizados con la colección de iniciativas parlamentarias son muy exhaustivos y enseñan

que el Naïve Bayes o incluso la SVM lineal (el "estado del arte" en clasificación documental) son claramente sobrepasados por nuestro modelo basado en redes bayesianas. Además, el capítulo ha presentado el problema de indexación automática basada en tesauros, que es –en nuestra opinión– un logro por sí mismo, y abre un nuevo subárea de investigación dentro de la clasificación documental.

Los resultados en clasificación de documentos estructurados, presentados en el capítulo 5, son de diversa consideración. Lo primero de todo, nuestra aproximación al problema de la clasificación XML parece ser útil, pero dependiente del modelo (el clasificador puerta OR no se beneficia de la solución "text replication" a pesar de que el Naïve Bayes sí lo hace). Para el problema de clasificación basada en enlaces, podemos afirmar que la lección principal aprendida es que las etiquetas de los documentos vecinos ayudan a clasificar el documento actual. En ese problema, hemos probado un enfoque basado en redes bayesianas con estructura fija, y otro con la estructura aprendida, siendo el segundo el más prometedor (aunque el primero también era capaz de mejorar un modelo básico). En este tema nuestra aportación es una nueva solución al problema y, de nuevo, en la que utilizamos una técnica distinta (redes Bayesianas, en nuestro caso).

Lista de Publicaciones

El contenido principal de esta tesis se ha presentado en el siguiente conjunto de publicaciones:

1. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero: *Automatic Indexing from a Thesaurus Using Bayesian Networks: Application to the Classification of Parliamentary Initiatives*. En: Khaled Mellouli (Ed.): Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 9th European Conference, ECSQARU 2007, Hammamet, Tunisia, October 31 - November 2, 2007, Proceedings. Lecture Notes in Computer Science **4724**, pp. 865–877, Springer 2007, ISBN 978-3-540-75255-4.

2. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Probabilistic Methods for Structured Document Classification at INEX'07*. En:

Norbert Fuhr, Jaap Kamps, Mounia Lalmas, Andrew Trotman (Eds.): Focused Access to XML Documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle, Germany, December 17-19, 2007. Selected Papers. Lecture Notes in Computer Science **4862**, pp. 195–206, Springer 2008, ISBN 978-3-540-85901-7.

3. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Or gate Bayesian networks for text classification: A discriminative alternative approach to multinomial naive Bayes*, actas del XIV Congreso Español sobre Tecnologías y Lógica Fuzzy, ESTYLF 2008, Mieres-Langreo (España), 17–19 Septiembre 2008.

4. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Thesaurus-based Automatic Indexing*. In: Handbook of Research on Text and Web Mining Technologies. Chapter XX, Vol. I, M. Song, Y.F. Brook Wu (Eds.), pp. 331–345, 2009. IGI Global. (ISBN 978-1-59904-990-8).

5. L. M. de Campos, A. E. Romero, *Bayesian network models for hierarchical text classification from a thesaurus*. Int. J. Approx. Reasoning **50**(7): 932-944 (2009).

6. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Probabilistic Methods for Link-Based Classification at INEX 2008*. En: Shlomo Geva, Jaap Kamps, Andrew Trotman (Eds.): Advances in Focused Retrieval, 7th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2008, Dagstuhl Castle, Germany, December 15-18, 2008. Revised and Selected Papers. Lecture Notes in Computer Science **5631**, pp. 453–459, Springer 2009, ISBN 978-3-642-03760-3.

7. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. R. Masegosa, A. E. Romero, *Link-based document classification using Bayesian Networks*. Proceedings of INEX 2009 (aparecerá), 2009.

Las siguientes referencias son publicaciones del autor que tratan de temas no incluidos en esta memoria:

8. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Un Prototipo de Motor de Búsqueda para los Diarios de Sesiones del Parlamento de Andalucía basado en Modelos Gráficos Probabilísticos*, Actas de las VI Jornadas de Transferencia Tecnológica de Inteligencia Artificial (TTIA'2005), 37-44, ISBN: 84-9732-435-8, 2005.

9. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Garnata: An Information Retrieval System for Structured Documents based on Probabilistic Graphical Models*, Proc. of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2006), 1024–1031, París (Francia).

10. A. E. Romero, *Geometry and Information Retrieval*, International Seminar on Applied Geometry in Andalusia, Granada 2006 (Satélite del ICM, Madrid 2006).

11. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *Garnata Implementing the SID and CID Models at INEX'06.* En: Norbert Fuhr, Mounia Lalmas, Andrew Trotman (Eds.): Comparative Evaluation of XML Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, December 17-20, 2006, Revised and Selected Papers. Lecture Notes in Computer Science **4518**, pp. 165–177, Springer 2007, ISBN 978-3-540-73887-9.

12. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. E. Romero, *A flexible object-oriented system for teaching and learning structured IR.* Proc. of the First International Workshop on Teaching and Learning of Information Retrieval (TLIR 2007), pp. 32–38, 2007.

13. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, C. Martín, A. E. Romero, *An Information Retrieval System for Parliamentary XML Documents based on Probabilistic Graphical Models.* En: O. Pourret, P. Naim, B. Marcot (Eds.), Bayesian Networks: A practical guide to applications, pp. 203–223, John Wiley & sons, ISBN: 978-0-470-06030-8, 2008.

14. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, C. Martín, A. E. Romero, *The Garnata Information Retrieval System at INEX'07*. En: Norbert Fuhr, Jaap Kamps, Mounia Lalmas, Andrew Trotman (Eds.): Focused Access to XML Documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle, Germany, December 17-19, 2007. Selected Papers. Lecture Notes in Computer Science **4862**, pp. 57–69, Springer 2008, ISBN 978-3-540-85901-7.

15. L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, C. J. Martín-Dancausa, A. E. Romero, *New Utility Models for the Garnata Information Retrieval System at INEX'08*. En: Shlomo Geva, Jaap Kamps, Andrew Trotman (Eds.): Advances in Focused Retrieval, 7th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2008, Dagstuhl Castle, Germany, December 15-18, 2008. Revised and Selected Papers. Lecture Notes in Computer Science **5631**, pp. 39–45, Springer 2009, ISBN 978-3-642-03760-3.

Trabajos futuros

Presentamos aquí, en esta sección, algunas ideas para continuar desarrollando este trabajo en el futuro. En primero lugar, para el clasificador basado en puertas OR, pensamos que podrían probarse diferentes esquemas de entrenamiento de los pesos. De hecho, hemos comprobado que los métodos que realizan una minimización de la verosimilitud condicional [63; 129], aunque son computacionalmente complejos, trabajan relativamente bien, incluso llegando a los niveles de la SVM lineal. Tal vez una metodología híbrida para el entrenamiento de pesos pudiera mantener un proceso ligero en la fase de entrenamiento (esto es, que no introdujera demasiada complejidad) y mejorase los resultados en la de clasificación. Sin duda, la cuestión que surge es la de si podemos aproximarnos al poder clasificador del algoritmo de la SVM lineal, que es el "estado del arte" en clasificación documental, y que da resultados sobresalientes en todas las colecciones de prueba. Para extender nuestro modelo, también pensamos que podríamos usar otros modelos

canónicos [42] (como, por ejemplo la puerta MAX ruidosa, la puerta AND, o una combinación entre diferentes modelos).

El trabajo en clasificación basada en tesauros podría extenderse para tratar con las particularidades de algunos tesauros muy específicos. Por ejemplo, en el caso del tesauro médico, MeSH [55], nuestro modelo podría ser mejorado con hacer tan sólo unos pocos cambios, para representar algunas pequeñas diferencias que este tesauro tiene, comparada con otros. Además, se podría hacer una experimentación más extensiva en una colección mayor (como, por ejemplo, el corpus Acquis, etiquetado con el tesauro Eurovoc). Con esta experimentación se podría desarrollar un modelo de entrenamiento para los diferentes parámetros (pesos), para mejorar nuestros resultados de clasificación y hacer un algoritmo libre de valores especificados manualmente.

Finalmente, la lista de contribuciones desarrolladas para clasificación estructurada se podría extender de varias formas. Para el problema de la clasificación XML, creemos que podría ser interesante una forma de obtener valores óptimos de replicación (en el esquema “text replication), para establecer conclusiones más sólidas en este campo. Para el problema de clasificación basada en enlaces, hay bastante posibilidad de mejora, sobre todo en el modelo en el que la red bayesiana se aprendía, debido a que se podrían probar otros algoritmos de aprendizaje y propagación, combinadas con una representación más rica del vecindario del documento (en vez de tan sólo vectores binarios).

References

[1] S. Acid, L.M. de Campos, and J.G. Castellano. Learning Bayesian network classifiers: searching in a space of acyclic partially directed graphs. *Mach. Learn.* **59**(3), 213–235, 2005.

[2] G. Adami, P. Avesani, and D. Sona. Clustering documents in a web directory. In *Proceedings of the WIDM-03, 5th ACM International Workshop on Web Information and Data Management*, 66–73, ACM Press, New York, US, 2003.

[3] G. Adami, P. Avesani and D. Sona. Clustering documents into a web directory for bootstrapping a supervised classification. *Data Knowl. Eng.* **54**(3), 301–325, 2005.

[4] *AGROVOC, multilingual agricultural thesaurus.* World Agricultural Information Center. Available at http://www.fao.org/scripts/agrovoc/frame.htm, 1998.

[5] C. Apté, F. Damerau, and S.M. Weiss. Automated learning of decision rules for text categorization. *ACM T. Inform. Syst.* **12**(3), 233–251, 1994.

[6] M. W. Berry (Ed.). *Survey of Text Mining: Clustering, Classification, and Retrieval.* Springer, New York, 2003.

[7] M. W. Berry, and M. Browne. *Understanding Search Engines. Mathematical Modeling and Text Retrieval.* Second Edition, SIAM Book Series: Software, Environments, and Tools, Philadelphia, 2005.

[8] B. E. Boser, I. Guyon, and V. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory.*, 144–152, 1992

[9] A. Bratko, and B. Filipic. Exploiting structural information for semi-structured document categorization. *Inf. Process. Manage.* **42**(3), 679–694, 2006.

[10] J. D. Brutlag, and C. Meek. Challenges of the email domain for text classification, In *Proceedings of ICML'00, 17th International Conference on Machine Learning,* 103–110, Morgan Kaufmann Publishers Inc, 2000.

[11] W. L. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE T. Knowl. Data En.* **8**, 195–210, 1996.

[12] C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Min. Knowl. Discov.* **2**(2), 121–167, 1998.

[13] A. Cano, S. Moral, and A. Salmerón. Algorithms for approximate probability propagation in Bayesian networks, In *Advances in Bayesian Networks, Studies in Fuzziness and Soft Computing.* **146**, 77–99, Springer-Verlag, 2004.

[14] L. M. de Campos. Modelos de Recuperación de Información Basados en Redes de Creencia, *Memoria de Habilitación*, 2003.

[15] L. M. de Campos, J. M Fernández-Luna, and J. F. Huete. The BNR model: foundations and performance of a Bayesian network-based retrieval model. *Int. J. Approx. Reason.* **34**, 265–285, 2003.

[16] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and A. E. Romero. Probabilistic Methods for Structured Document Classification at INEX'07. In *Proceedings of the INEX 2007, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval,* 195–206, 2007.

[17] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and A. E. Romero. Automatic indexing from a thesaurus using Bayesian networks: Application to the classification of parliamentary initiatives. In *Proceedings of ECSQARU 2007, 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty,* 865–877, 2007.

[18] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and A. E. Romero. OR gate Bayesian networks for text classification: a discriminative alternative approach to multinomial naive Bayes, *Actas del XIV Congreso Español sobre Tecnologías y Lógica Fuzzy*, 385–390, 2008.

[19] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and A. E. Romero. Probabilistic methods for link-based classification at INEX'08, In *Proceedings of the INEX 2008, 7th International Workshop of the Initiative for the Evaluation of XML Retrieval,* 453–459, 2009.

[20] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, A. R. Masegosa, and A. E. Romero. Link-based document classification using Bayesian Networks. To appear in *Proceedings of the INEX 2009, 8th International Workshop of the Initiative for the Evaluation of XML Retrieval,* INEX 2009: 398–407, 2009.

[21] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and A. E. Romero. Thesaurus based automatic indexing. In *Handbook of Research on Text and Web Mining Technologies, Chapter XX, Vol. I, M. Song, Y.F. Brook Wu (Eds.),* IGI Global, 331–345, 2009.

[22] L. M. de Campos, and A. E. Romero. Bayesian network models for hierarchical text classification from a thesaurus *Int. J. Approx. Reason.* **50**(7), 932–944, 2009.

[23] V. R. de Carvalho, and W. W. Cohen. On the collective classification of email "speech acts", In *Proceedings of the SIGIR 2005, 28th annual international ACM SIGIR conference on Research and development in information retrieval,* 345–352, 2005.

[24] W. W. Cohen, and Y. Singer. Context-sensitive learning metods for text categorization. In *Proceedings of the SIGIR'96, 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 307–315, 1996.

[25] C. Castillo, B. D. Davison, L. Denoyer, and P. Gallinari (Eds.). Proceedings of GRAPHLAB'07, ECML/PKDD 2007 Workshop on Graph Labeling, 2007 (available at http://www.ecmlpkdd2007.org/CD/workshops/GraphLab/workshop_print.pdf).

[26] E. Castillo, J. M. Gutiérrez, and A. S. Hadi. **Expert Systems and Probabilistic Network Models**, Springer-Verlag, New York, 1998.

[27] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data,* 307–318, 1998.

[28] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Using taxonomy, discriminants, and signatures for navigating in text databases. In *Proceedings of the VLDB'97, 23rd International Conference on Very Large Data Bases*, 446–455, 1997.

[29] D. M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian Netoworks is NP-Hard, *Technical Report MSR-TR-94-17*, 1994.

[30] G. F. Cooper. The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks. *Artif. Intell.* **42**, 393–405, 1990.

[31] C. Cortes, and V. Vapnik. Support-Vector Networks. *Mach. Learn.* **20**(3), 273–297, 1995.

[32] F. Crestani, L. M. de Campos, J. M. Fernández-Luna, and J. F. Huete. A Multi-layered Bayesian Network Model for Structured Document Retrieval. In *Proceedings of ECSQARU 2003, 7th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty,* 74–86, 2007.

[33] P. Dagum, and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artif. Intell.* **60**, 141–153, 1993.

[34] F. Debole, and F. Sebastiani. An analysis of the relative hardness of Reuters-21578 subsets. *J. Am. Soc. Inf. Sci. Tec.* **56**(6), 584–596, 2005.

[35] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. Roy. Stat. Soc. B. Met.* **39**(1), 1–38, 1977.

[36] L. Denoyer, and P. Gallinari. Bayesian network model for semi-structured document classification. *Inf. Process. Manag.* **40**(5), 807–827, 2004.

[37] L. Denoyer, and P. Gallinari. The Wikipedia XML Corpus. *SIGIR Forum.* **40**(1), 64–69, 2006.

[38] L. Denoyer, and P. Gallinari Overview of the INEX 2008 XML Mining Track,
Lecture Notes in Computer Science 5631:401–411, 2009.

[39] L. Denoyer, and P. Gallinari. Report on the XML Mining Classification Track at INEX 2009, INEX 2009 Pre 339–342, 2009. Overview of the INEX 2009 XML Mining Track, INEX 2009:401–411, 2009.

[40] DESY, The high energy physics index keywords, available at http://www-library.desy.de/schlagw2.html, 1996.

[41] S. J. DeRose. Navigation, Access, and Control Using Structured Data. *Am. Archivist.* **60**(33), 298–30, 1997.

[42] F. J. Díez, and M. J. Druzdzel. Fundamentals of canonical models. In IX Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA-TTIA 2001), 1125–1134, Gijon, Spain, 2001.

[43] F. J. Díez, and M. J. Druzdzel. Canonical probabilistic models for knowledge engineering. Technical Report CISIAD-06-01. UNED, Madrid, 2006.

[44] R. Diestel. *Graph Theory,* Springer-Verlag, Heidelberg, Graduate Texts in Mathematics, **173**, 2005.

[45] S. T. Dumais, and H. Chen. Hierarchical classification of web document. In *Proceedings of the 23th ACM International Conference on Research and Development in Information Retrieval*, 256–263, 2000.

[46] S. T. Dumais, J. C. Platt, D. Heckerman, and M. Sahami. *Inductive Learning Algorithms and Representations for Text Categorization*, CIKM 1998: 148–155, 1998.

[47] Elvira Consortium. Elvira: An environment for probabilistic graphical models, Proceedings of the First European Workshop on Probabilistic Graphical Models, 222–230, 2002. Elvira software available at http://leo.ugr.es/~elvira.

[48] EUROVOC. Thesaurus Eurovoc - Volume 2: Subject-Oriented Version. Ed. 3/English Language. Annex to the index of the Official Journal of the EC. Luxembourg, Office for Official Publications of the European Communities. Available at http://europa.eu.int/celex/eurovoc, 1995.

[49] P. Gallinari, L. Denoyer, and A. E. Romero. Bibliography on Methods for Collective Classification on the Framework of Markov networks, Unpublished Tech Report, available at http://decsai.ugr.es/~aeromero/reportCollective.pdf

[50] A. Genkin, D. D. Lewis, and D. Madigan. Large-Scale Bayesian Logistic Regression for Text Categorization, *Technometrics.* **49**(3), 291–304, 2007.

[51] K. Golub. Automated subject classification of textual web documents. *J. Doc.* **62**, 350–371, 2006.

[52] P. J. Hayes, and S. P. Weinstein. CONSTRUE/TIS: A system for content-based indexing of a database of news stories. In Proc. 2nd Conf. on Innovative Applications of AI (IAAI), 49–64. AAAI Press, 1991.

[53] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: the combination of knowledge and statistical data. *Mach. Learn.* **20**(3), 197–243, 1995.

[54] W. R. Hersh. Information Retrieval: A Health and Biomedical Perspective, 2nd Edition, Springer, New York, 2003.

[55] W. R. Hersh, C. Buckley, T. J. Leone, and D. Hickman. *Ohsumed: an interactive retrieval evaluation and new large test collection for research.* Proc. of the ACM SIGIR Conference, 1994.

[56] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In ACM SIGKDD'04, 593–598, New York, NY, USA, 2004. ACM.

[57] F. V. Jensen. *An Introdution to Bayesian Networks*, UCL Press, 1996.

[58] F. V. Jensen. *Bayesian Networks and Decision Graphs*, Springer-Verlag, New York, 2001.

[59] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, 143–151, 1997.

[60] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, 200–209, 1998

[61] T. Joachims. *A Statistical Learning Model of Text Classification for Support Vector Machines*. SIGIR 2001: 128–136, 2001.

[62] T. Joachims. SVM Light Support Vector Machine, 2002, available at http://svmlight.joachims.org.

[63] R. Jurgelenaite, and T. Heskes. Learning symmetric causal independence models. *Mach. Learn.* **71**, 133–153, 2008.

[64] S: Kim, K. Han, H. Rim, and S. Myaeng. Some Effective Techniques for Naive Bayes Text Classification. *IEEE T. Knowl. Data En.* **18**(11), 1457–1466, 2006.

[65] B. Klimt, and Y. Yang. (2004). The Enron corpus: A new dataset for email classification research. In Proceedings of ECML-04, 15th European Conference on Machine Learning, 217226.

[66] M. A. Kłopotek. Very large Bayesian multinets for text classification. *Future Gener. Comp. Sy.* **21**(7), 1068–1082, 2005.

[67] D. Koller, and M. Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Learning*, 170–178, 1997.

[68] C. H. A. Koster, and J. Beney. *On the Importance of Parameter Tuning in Text Categorization*. Ershov Memorial Conference 2006: 270-283

[69] S. Kullback. *Information Theory and Statistics,* Dover Publications, 1968.

[70] H. Marjorie, and R. Hainebach. Multilingual Machine Indexing. NIT1996. Available at http://joan.simmons.edu/~chen/nit/NIT'96/96-105-Hava.html, 1996.

[71] K. Lang. NEWSWEEDER: learning to filter netnews. In Proceedings of ICML-95, 12th International Conference on Machine Learning (Lake Tahoe, CA, 1995), 331–339.

[72] L. S. Larkey, and W. B. Croft. Combining classifiers in text categorization. In: *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 289–297 (1996).

[73] R. R. Larson. Experiments in automatic library of congress classification. *J. Am. Soc. Inform. Sci.* **43**(2), 130–148, 1992.

[74] B. Lauser, and A. Hotho. Automatic multi-label subject indexing in a multilingual envirinment. *Lecture Notes in Computer Science*, 2769:140–151, 2003.

[75] D. D. Lewis, and W. Gale. A sequential algorithm for training text classifiers. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval,* 3–12, 1994.

[76] D. D. Lewis. Naïve (Bayes) at Forty: The Independence Assumption in Information Retrieval, In *Proceedings of ECML'98, 10th European Conference On Machine Learning,* 1998. 4–15,

[77] D. D. Lewis, and M. Ringuette. A Comparison of Two Learning Algorithms for Text Categorization, In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval,* 81–93, 1994.

[78] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. *J. Mach. Learn. Res.* **5**, 361–397, 2004.

[79] Q. Lu and L. Getoor. Link-based Classification. ICML 2003: 496–503

[80] S. A. Macskassy, and F. J. Provost. Simple Relational Classifier, In *Proceedings of the MRDM-2003, 2nd Workshop on Multi-Relational Data Mining, at KDD-2003,* 64–76, 2003.

[81] S. A. Macskassy, and F. Provost. Classification in Networked Data: A toolkit and a univariate case study. To appear in *J. Mach. Learn. Res.*, 2010.

[82] F. Maes, Stéphane Peters, L. Denoyer, and P. Gallinari. Simulated Iterative Classification: A New Learning Procedure for Graph Labeling. In *Proceedings of ECML PKDD 2009, European Conference on Machine Learning and Knowledge Discovery in Databases,* 47–62, 2009.

[83] M. E. Maron. Automatic indexing: An experimental inquiry. *J. ACM.* **8**, 404–417, 1961.

[84] I. Martínez, C. Rodríguez, and A. Salmerón. Dynamic importance sampling in Bayesian networks using factorisation of probability trees. *Proceedings of PGM'06, Third European Workshop on Probabilistic Graphical Models,* 187–194, 2006.

[85] A. McCallum, and K. Nigam. A Comparison of event models for Naive Bayes text classification. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization,* 1998.

[86] L. K. McDowell, K. M. Gupta, and D. W. Aha. Cautious Inference in Collective Classification. AAAI 2007: 596–601

[87] O. Medelyan. Automatic Keyphrase Indexing with a Domain-Specific Thesaurus. MSc. Thesis, 2005.

[88] O. Medelyan, and I. H. Witten. Thesaurus-based index term extraction for agricultural documents, In *Proceedings of the 6th Agricultural Ontology Service (AOS) workshop at EFITA/WCCA 2005*, 2005.

[89] O. Medelyan, and I. H. Witten. Thesaurus based automatic keyphrase indexing, In *Proceedings of the JCDL 2006*, 296–297, 2006.

[90] A. Miles, and D. Brickley. *SKOS Core Guide*, W3C Working Draft available at http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20051102, 2005.

[91] T. P. Minka. Empirical Risk Minimization is an incomplete inductive principle, *unpublished MIT Media Lab note,* available at http://research.microsoft.com/en-us/um/people/minka/papers/erm.html.

[92] A. Montejo-Ráez. A. Towards conceptual indexing using automatic assignment of descriptors Proceedings of the Workshop on Personalization Techniques. In *Proceedings of Electronic Publishing on the Web, at the 2nd International Conference on Adaptive Hypermedia and Adaptive Web.*, Málaga, Spain, 2002.

[93] A. Montejo-Ráez. Automatic Text Categorization of documents in the High Energy Physics domain, PhD. Thesis, Univ. de Granada, 2005.

[94] S. Moral, and A. Salmerón. Dynamic importance sampling in Bayesian networks based on probability trees. *Int. J. Approx. Reason.*, **38**(3), 245–261, 2005.

[95] A. Moschitti. A Study on Optimal Parameter Tuning for Rocchio Text Classifier. In *Proceedings of the ECIR 2003,* 420–435, 2003.

[96] R. Moskovitch, S. Cohen-Kashi, U. Dror, and I. Levy. Multiple hierarchical classification of free-text clinical guidelines. *Artif. Intell. Med.*. **37**(3), 177–190, 2006.

[97] I. Moulinier, G. Raskinis, and J. Ganascia. Text categorization: A symbolic approach. In: Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval, 1996.

[98] NAL Thesaurus, available on http://agclass.nal.usda.gov/agt/agt.shtml, 2002.

[99] R. Neapolitan. Learning Bayesian Networks, Prentice Hall, 2003.

[100] J. Neville, and D. Jensen. Iterative classification in relational data, In *Proceedings of the AAAI Workshop on Learning Statistical Models from Relational Data,*, 2000.

[101] National Library of Medicine. *Medical Subject Headings.* Bethesda, Maryland, USA, 1986.

[102] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan and Kaufmann, San Mateo, 1988.

[103] J. Platt. Probabilistic outputs for Support Vector Machines and comparisons to regularized likelihood methods, in *Advances in Large Margin Classifiers,* 61–74, MIT Press, 1999.

[104] M. F. Porter. Snowball Package, available at http://snowball.tartarus.org.

[105] M. F. Porter. An Algorithm for Suffix Stripping, **Program. 14**(3): 130–137, 1980.

[106] J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger. Tackling the Poor Assumptions of Naive Bayes Text Classifiers, In *Proceedings of the ICML'03, Twentieth International Conference on Machine Learning,* 2003, 616–623

[107] A. Renyi. *Foundations of Probability,* San Francisco, CA: Holden-Day, 1970.

[108] C. J. van Rijsbergen. *Information Retrieval,* (Second edition), Butter Worths, London, 1979.

[109] S. E. Robertson, and K. Spärck-Jones. Relevance weighting of search terms. *J. Am. Soc. Inform. Sci.* **27**, 129–146, 1976.

[110] J. J. Rocchio. Relevance feedback in information retrieval. In G.Salton (Ed.), *The SMART Retrieval System: Experiments in Automatic Document Processing,* 313–323. Prentice Hall, 1971.

[111] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Cornell Aeronautical Laboratory, Psychological Review.* **65**(6), 386–408, 1958.

[112] M. Ruiz, and P. Srinivasan. Hierarchical text categorization using neural networks. *Inform. Retrieval.*, **5**(1), 87–118, 2002.

[113] M. Sahami. *Learning Limited Dependence Bayesian Classifiers*, In *Proceedings of the KDD-96, Second International Conference on Knowledge Discovery and Data Mining,* 335–338, 1996.

[114] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian approach to filtering junk e-mail, in *Learning for Text Categorization: Papers from the 1998 Workshop, Madison, Wisconsin,* AAAI Technical Report WS-98-05, 1998.

[115] G. Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Commun. ACM.* **18**(11), 613–620, 1975.

[116] G. Salton, and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Inf. Process. Manage.* **24**(5), 513–523, 1988.

[117] G. Salton, and M. E. Lesk. The SMART automatic document retrieval systemsan illustration. *Commun. ACM* **8**(6), 391–398, 1965.

[118] F. Sebastiani, and E. Gabrilovich. *Bibliography on Automated Text Categorization*, available at http://liinwww.ira.uka.de/bibliography/Ai/automated.text.categorization.html.

[119] F. Sebastiani. Machine learning in automated text categorization, *ACM Comput. Surv.* **34**(1), 1–47, 2002.

[120] P. Sen, and Getoor. Empirical Comparison of Approximate Inference Algorithms for Networked Data, In *Proceedings of the ICML Workshop on Open Problems in Statistical Relational Learning,* 2006.

[121] P. Sen, and L. Getoor. Link-based Classification, *Technical Report CS-TR-4858, University of Maryland, Number CS-TR-4858*, 2007.

[122] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective Classification in Network Data, *Technical Report CS-TR-4905 and UMIACS-TR-2008-04*, 2008.

[123] C. E. Shannon. *A mathematical theory of communication*, *AT&T Tech J.*, **27**, 1948. Reprinted in: *Mobile Computing and Communications Review* **5**(1), 3–55, 2001.

[124] R. Steinberger. Using Thesauri for Information Extraction and for the Visualisation of Multilingual Document Collections. In *Proceedings of the OntoLex2000, Workshop on Ontologies and Lexical Knowledge Bases*, 130–141, 2000.

[125] R. Steinberger. Cross-lingual keyword assignment. In *Proceedings of the SEPLN2001, XVII Conference of the Spanish Society for Natural Language Processing,* 273–280, 2001.

[126] R. Steinberger, B. Pouliquen, and J. Hagman. Cross-lingual Document Similarity Calculation Using the Multilingual Thesaurus Eurovoc. In *Computational Linguistics and Intelligent Text Processing, Third International Conference, CICLing'2002.*, 415–424, Mexico-City, Mexico, 2002.

[127] R. Steinberger, B. Pouliquen, and C. Ignat. Automatic Annotation of Multilingual Text Collections with a Conceptual Thesaurus, In *Workshop in Ontologies and Information Extraction (EUROLAN2003),* 2003.

[128] V. N. Vapnik. *The Nature of Statistical Learning Theory,* Berlin: Springer-Verlag, 1995.

[129] J. Vomlel. Noisy-or classifier. *Int. J. Intell. Syst.* **21**(3), 381–398, 2006.

[130] S. M. Weiss, C. Apté, F. J. Damerau, D. E. Johnson, F. J. Oles, T. Goetz, and T. Hampp. Maximizing text-mining performance. *IEEE Intell. Syst.* **14**(4), 63–69, 1999.

[131] I. H. Witten, and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*, Second Edition, Morgan Kaufmann, 2005.

[132] E. Wiener, J. O. Pedersen, and A. S. Weigend. A neural network approach to topic spotting. In *Proceedings of the SDAIR95, Fourth Annual Symposium on Document Analysis and Information Retrieval.*, 1995.

[133] Y. Yang, and C. G. Chute. An example-based mapping method for text categorization and retrieval. *ACM T. Inform. Syst.*, **12**(3), 253–277, 1994.

[134] Y. Yang. An evaluation of statistical approaches to MEDLINE indexing. In *Proceedings of the AMIA Annu Fall Symp.*, 358–362, 1996.

[135] Y. Yang. An evaluation of statistical approaches to text categorization. *Inform. Retrieval.*, **1**, 69–90, 1999.

[136] Y. Yang. A study of thresholding strategies for text categorization, In *Proceedings of the SIGIR'01, 24th annual international ACM conference on Research and Development in Information Retrieval,* 137–145, 2001.

[137] Y. Yang, and S. Slattery. A study of approaches to hypertext categorization. *J. Intell. Inf. Syst.* **18**, 219–241, 2002.

[138] J. Yi, and N. Sundaresan. A classifier for semi-structured documents. In *Proceedings of the SIGKDD'00, 6th ACM International Conference on Knowledge Discovery and Data Mining,* 340–344. ACM Press, 2000.

[139] J. Yedidia, W. T.Freeman, and Y. Weiss. *Generalized belief propagation.* In *Proceedings of NIPS'00, Advances in Neural Information Processing Systems 13,*, 689–695, 2000.