UNIVERSIDAD DE GRANADA

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA INFORMÁTICA

Departamento de Ciencias de la Computación
e Inteligencia Artificial

DESARROLLO Y VALIDACIÓN DE MODELOS PARA EL APRENDIZAJE DE LA ORDENACIÓN EN RECUPERACIÓN DE INFORMACIÓN

Memoria que presenta

en opción al grado de
Doctor en Informática

AUTOR:
Oscar José Alejo Machado

DIRECTORES:
Dr. Juan Manuel Fernández Luna
Dr. Juan Francisco Huete Guadix

Granada, Junio de 2014

Editor: Editorial de la Universidad de Granada
Autor: Óscar José Alejo Machado
D.L.: GR 2123-2014
ISBN: 978-84-9083-143-4

La memoria titulada **Desarrollo y Validación de Modelos para el Aprendizaje de la Ordenación en Recuperación de Información**, que presenta D. Oscar José Alejo Machado, profesor de la Universidad de Cienfuegos (Cuba), para optar por el grado de DOCTOR, ha sido realizada bajo la dirección de Juan Manuel Fernández Luna y Juan Francisco Huete Guadix, Profesores Titulares del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada (España).

Granada, Mayo de 2014.

Los directores

Fdo.: Juan Manuel Fernández Luna

Fdo.: Juan Francisco Huete Guadix

El doctorando

Fdo.: Oscar José Alejo Machado

El doctorando Oscar José Alejo Machado y los directores de la tesis Juan Manuel Fernández Luna y Juan Francisco Huete Guadix garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

En Granada, a 13 de mayo de 2014

Director/es de la Tesis | Doctorando

Fdo.: | Fdo.:

A la memoria de mi padre Oscar Armando Alejo Cruz,
a mi madre Marlén C. Machado y a mi hermano Enmanuel.

"Lo que hace crecer el mundo, no es saber como está hecho
sino el esfuerzo por descubrirlo".
(José Martí)

Agradecimientos

Me gustaría hacer constar mi agradecimiento a todas las personas e instituciones que en esta etapa me apoyaron.

A mis grandes tutores, que en Cuba los bautizamos como los "famosos Juanes", por esa mezcla idónea de intelecto y simpatía. A ellos les debo el haber transitado por este largo camino, pues fueron mis guías, mis ejemplos y al final se han convertido también en mi familia.

A Rafael Bello por ser nuestro líder y amigo. Estando siempre dispuesto cuando lo necesitaba.

A Jose Luis Verdegay le estaré eternamente agradecido por su interés constante en nuestro desarrollo profesional.

A todos los profesores de España y Cuba que compartieron conmigo sus conocimientos en esta segunda edición del Programa Doctoral Iberoamericano en Soft Computing.

A Eduardo Concepción por ser un amigo entrañable, consejero y colaborador en tiempos difíciles.

A Yuniol mi compañero y amigo de siempre, por su apoyo incondicional.

A todos los profesores del departamento de informática y matemática de la Universidad de Cienfuegos. En especial a Roberto Suárez y Miguel Santana por su interés constante y aportes a mi investigación.

A Ramiro, Marilyn, Leticia y profesores del CEI y de la Facultad de Matemática y Computación de la UCLV por sus consejos y atenciones.

A todos los profesores del DECSAI que de una forma u otra contribuyeron a mi investigación.

A todos mis compañeros del Programa Doctoral Iberoamericano en Soft Computing tanto de Cuba como del extranjero, por su confianza y su apoyo absoluto.

A mi gran familia de España Juan Luis, Pili, Pablo, Diana, Alejandro, en fin a todos, por su apoyo incondicional.

A mis hermanos de fe Nicolás y Abi por estar siempre a mi lado y darme consejos oportunos.

A todos mis alumnos de informática que durante estos años me han estimulado a seguir en esta labor investigativa.

Al Programa de Becas Erasmus Mundus - ánimo, Chévere! por facilitarme una estancia en la UGR durante el período del 1 de septiembre de 2010 al 24 de febrero de 2011.

Al Programa de Becas para Jóvenes Investigadores del Grupo Coimbra durante el período del 1 de marzo al 31 de mayo de 2014.

Al proyecto "Granada Excellence Network of Innovation Laboratories"(GENIL) de la UGR por su apoyo económico para llevar a términos la defensa de esta tesis.

A la Asociación Universitaria Iberoamericana de Postgrado (AUIP), a la Consejería Española de Innovación, Ciencia y Empresa de la Junta de Andalucía y al Ministerio de Ciencia e Innovación, por la financiación y apoyo económico que me han brindado durante todo este tiempo, mediante los proyectos TIN2008-06566-C04-01, TIN2011-28538-C02-02, P09-TIC-4526 y el Proyecto de Excelencia TIC-04526.

Y finalmente, para que se lleve todo el peso del mundo, al apoyo incondicional, amor y comprensión de toda mi familia que me ha dado las fuerzas necesarias hasta el último segundo.

Y si alguien se me ha quedado sin mencionar, pues no lo tome en cuenta que en mi corazón sí lo llevo grabado.

A todos, muchísimas gracias...

Producción Científica del autor

Artículos en Revistas

- *"RankPSO: A New L2R algorithm Based on Particle Swarm Optimization"*. Oscar J. Alejo Machado; Juan M. Fernández-Luna; Juan F. Huete-Guadix. Journal of Multiple-Valued Logic and Soft Computing (JMVLSC), vol. 23, no.1-2, pp.1-34, (2014) ISSN: 1542-3980. JCR: 1.047. (LOGIC Q1 2/19, COMPUTER SCIENCE, THEORY & METHODS Q2 32/100).

- *"Fisherman Search Procedure"*. Oscar J. Alejo Machado; Juan M. Fernández-Luna; Juan F. Huete-Guadix. Journal Progress in Artificial Intelligence. Special Issue: Metaheuristics Applied To Solve Real World Problems (Trabajo seleccionado de la CAEPIA). ISSN: 2192-6352 (2014). DOI 10.1007/s13748-014-0052-7.

- *"Metric study of the scientific production about "Learning to Rank" from 2000 to 2013"*. Oscar J. Alejo Machado; Juan M. Fernández-Luna; Juan F. Huete-Guadix. Sometido a Scientometrics. ISSN: 0138-9130 (2014). JCR: 2.133 (INFORMATION SCIENCE & LIBRARY SCIENCE Q1 7/85).

Artículos en Congresos Internacionales

- *"L2RLab: Integrated Experimenter Environment for Learning to Rank"*. Oscar J. Alejo Machado; Juan M. Fernández-Luna; Juan F. Huete-Guadix; Eleazar Moreno-Cerrud. Flexible Query Answering Systems - 10th International Conference, FQAS 2013, Lecture Notes in Computer Science, Vol. 8132, Granada, Spain, September 18-20, 2013. p. 543-554. ISBN 978-3-642-40768-0, (CORE2013 C, 0801 - Artificial Intelligence and Image Processing), H.L. Larsen et al. (Eds.).

- *"Fisherman Search Procedure"*. Oscar J. Alejo Machado; Juan M. Fernández-Luna; Juan F. Huete-Guadix; Eduardo R. Concepción-Morales. Advances in Artificial Intelligence - IBERAMIA 2012 - 13th Ibero-American Conference on AI, IBERAMIA

2012 Lecture Notes in Computer Science, Vol. 7637, Cartagena de Indias, Colombia, November 13-16, 2012. p. 291-299. ISBN 978-3-642-34653-8, J. Pavón et al. (Eds.).

- *"Direct Optimization of Evaluation Measures in Learning to Rank using Particle Swarm"*. Oscar J. Alejo Machado; Juan M. Fernández-Luna; Juan F. Huete-Guadix; Ramiro Pérez-Vázquez. Proceedings - 21st International Workshop on Database and Expert Systems Applications, DEXA 2010 , pp. 42-46. ISBN 978-1-4244-8049-4.

Artículos en Congresos Nacionales

- *"Optimización de medidas de desempeño en el aprendizaje de la ordenación usando enjambres de partículas"*. Oscar J. Alejo Machado; Juan M. Fernández-Luna; Juan F. Huete-Guadix. VI Conferencia Científica de la Universidad de las Ciencias Informáticas UCIENCIA 2012, VI Taller De Inteligencia Artificial. ISBN 978-959-286-019-3.

- *"Estado del Arte del Aprendizaje de la Ordenación para la Recuperación de Información"*. Oscar J. Alejo Machado; Juan M. Fernández-Luna; Juan F. Huete-Guadix; Ramiro Pérez-Vázquez. Tendencias de Soft Computing, Editorial Feijóo, UCLV. 2009. ISBN-959250525-4.

Resumen

Learning to Rank (L2R) es un tópico vigente de investigación que emerge como intersección de las ramas de la Recuperación de Información (RI) y el *Machine Learning* (ML). El problema clave del L2R para la RI se basa en determinar el *ranking* ideal en el que deben ser mostrados los documentos recuperados ante una consulta de un usuario, considerando la relevancia de tales documentos con relación a un conjunto de características.

Diseñar funciones efectivas de *ranking* que tengan un impacto directo a la hora de mostrar los resultados de la búsqueda, es esencial para muchas aplicaciones dentro de la RI, el procesamiento del lenguaje natural y la minería de datos. Constituye una tarea compleja que ha atraído el interés de la comunidad científica y la incorporación constante de nuevas técnicas de ML. Específicamente, los nuevos métodos llevan a cabo un aprendizaje supervisado, encaminado a crear de forma automática un modelo de *ranking* usando datos de entrenamiento y medidas de evaluación utilizadas en RI.

A pesar de estos esfuerzos, todavía las soluciones sólo garantizan una aproximación local. Muchos algoritmos plantean problemas desde su concepción, pues conducen sus esfuerzos hacia modelos sesgados hacia las consultas más relevantes. En otros casos, la posición de los documentos en la lista ordenada es invisible a la función de pérdida que utilizan, también el convertir múltiples categorías ordenadas a preferencias a pares entre documentos conlleva a obviar información acerca de las relaciones más sensibles de relevancia y puede introducir instancias ruidosas. Por otro lado, optimizar directamente medidas de evaluación siendo estas discontinuas y no diferenciables resulta en una tarea no trivial, que lleva a problemas de complejidad algorítmica y a soluciones locales.

En la presente tesis, se describe el dominio conceptual del problema del L2R, su desglose categórico, principales métodos, medidas de evaluación y colecciones estándares para la experimentación. Se presenta un estudio bibliométrico de la producción científica del L2R que nos permitió conocer información fidedigna sobre el comportamiento y las tendencias investigativas dentro del L2R y así definir algunos patrones a seguir en esta investigación. Entonces, a partir de estas premisas, se introducen una serie de modelos y estrategias

para mejorar la tarea del L2R sobre diferentes colecciones de datos. Se propone un primer método basado en *Particle Swarm Optimization* (PSO) y que optimiza directamente cualquier medida de evaluación de RI. Se desarrolla una nueva metaheurística de optimización global, que es adaptada y mejorada para conformar un nuevo método de L2R. En estos métodos también se incorporan estrategias y nuevos métodos para la reducción de dimensionalidad de las colecciones de datos. Todos estos métodos y propuestas fueron validadas experimentalmente, demostrando mejoras sustanciales para la tarea del *ranking*. Finalmente, también se propone una herramienta software para asistir y facilitar la labor experimental de los investigadores del área.

Índice general

Resumen. ix

Índice de figuras. xv

Índice de cuadros. xviii

Índice de algoritmos. xxi

Notación. xxiii

PARTE I. Introducción General. 1

1. Introducción y motivación 1

2. Introducción al Aprendizaje de la Ordenación en Recuperación de Información 7

2.1. Introducción . . . 7

2.2. Modelos convencionales de RI . . . 7

2.3. Descripción del problema del L2R . . . 10

2.4. Formulación general del L2R . . . 15

2.5. Medidas de evaluación de RI utilizadas en L2R . . . 16

2.6. Principales categorías y modelos del L2R . . . 19

2.6.1. Enfoque por Puntos (*Pointwise Approach*) . . . 21

2.6.2. Enfoque por Pares (*Pairwise Approach*) . . . 24

2.6.3. Enfoque por Listas (*Listwise Approach*) . . . 26

2.6.4. Evolución histórica . . . 27

2.7. Optimización directa de medidas de rendimiento . . . 28

2.7.1. Minimización de funciones de pérdida que acoten superiormente a una función de pérdida básica definida sobre las medidas de RI . . . 29

2.7.2. Aproximación de medidas de rendimiento de RI mediante funciones fáciles de manipular . . . 31

2.7.3. Tecnologías especialmente diseñadas para optimizar medidas de rendimiento de RI no suaves 32
2.8. Colecciones estándar de prueba en L2R 34
2.9. Conclusiones 36

PARTE II. Estudio Bibliométrico sobre Learning to Rank. 37

3. Estudio bibliométrico de la producción científica sobre Aprendizaje de la Ordenación 39

3.1. Introducción 39
3.2. Los Estudios Métricos en las Ciencias de la Información 40
3.3. Scopus: Base de Datos bibliográfica en línea 42
3.4. Procedimiento general realizado para el análisis métrico 44
3.5. Indicadores Unidimensionales 45
3.5.1. Evolución de la temática "Learning to rank" 45
3.5.2. Temáticas más tratadas o exploradas en L2R 46
3.5.3. Áreas del conocimiento en las que más se aplica el L2R 47
3.5.4. Productividad científica de autores en el L2R 48
3.5.5. Impacto de la productividad de los autores en el L2R 48
3.5.6. Fuentes de información más relevantes en L2R 49
3.5.7. Países con mayor actividad científica en el L2R 51
3.6. Indicadores multidimensionales 51
3.6.1. Desglose grupal de las temáticas tratadas en L2R 51
3.6.2. Temáticas en auge y estables 53
3.6.3. Temáticas incipientes 56
3.6.4. Temáticas obsoletas o abandonadas 58
3.6.5. Temáticas más tratadas por los autores de mayor impacto 59
3.6.6. Temáticas más tratadas en las fuentes de información más relevantes 61
3.6.7. Colaboración de los autores que presentan más de 5 trabajos 63
3.7. Fundamentos para nuestra investigación 64
3.8. Conclusiones 65

PARTE III. Modelos y estrategias para Learning to Rank. 67

4. Aprendizaje de la Ordenación basado en Optimización con Enjambre de Partículas 69

4.1. Introducción . . . 69

4.2. Optimización con enjambre de partículas . . . 69

4.2.1. PSO Clásico . . . 70

4.2.2. Peso inercial . . . 73

4.3. Método RankPSO . . . 73

4.3.1. Descripción del algoritmo . . . 73

Ventajas del algoritmo: . . . 80

4.3.2. Reducción de dimensionalidad usando PCA y análisis de Clúster . . . 81

4.4. Experimentación . . . 86

4.4.1. Evaluación de rendimientos obtenidos y comparación de métodos . . . 86

4.4.2. Análisis estadístico . . . 92

4.4.3. Análisis del coste computacional . . . 95

4.5. Conclusiones . . . 97

5. Procedimiento de Búsqueda del Pescador: una nueva metaheurística de optimización global 99

5.1. Introducción . . . 99

5.2. Estrategia intuitiva de la pesca . . . 100

5.3. Descripción del algoritmo . . . 101

Ventajas del algoritmo: . . . 105

5.4. Parametrización . . . 106

5.4.1. Puntos de captura . . . 107

5.4.2. Número de iteraciones y lanzamientos . . . 107

5.4.3. Red de pesca y coeficiente de amplitud . . . 110

5.5. Experimentación . . . 112

5.5.1. Funciones estándares para las pruebas . . . 113

5.5.2. Metaheurísticas seleccionadas para la comparación . . . 113

5.5.3. Evaluación de rendimientos . . . 115

5.6. Conclusiones . . . 118

6. Aplicación del Procedimiento de Búsqueda del Pescador al Aprendizaje de la Ordenación 120

6.1. Introducción . . . 120

6.2. Método RankFSP . . . 120

Ventajas del algoritmo: . . . 127

6.3. Preprocesamiento de las colecciones de datos . . . 128

6.3.1. Justificación . . . 128

6.3.2. Selección de rasgos . . . 130

Propuesta de selección de rasgos. . . . 132

6.4. Experimentación . . . 135

6.4.1. Evaluación de rendimientos obtenidos y comparación de métodos . 136

6.4.2. Análisis estadístico . . . 143

6.4.3. Análisis del coste computacional . . . 146

6.5. Conclusiones . . . 149

PARTE IV. Herramienta software para la tarea del L2R. 151

7. L2RLab: Entorno Integrado de Experimentación para el Aprendizaje de la Ordenación 153

7.1. Introducción . . . 153

7.2. Justificación de la herramienta . . . 154

7.3. Diseño de experimentos . . . 156

7.3.1. Características . . . 156

7.3.2. Procedimiento . . . 157

7.4. Aspectos técnicos de L2RLab . . . 158

7.4.1. Framework . . . 159

7.4.2. Aplicación visual . . . 160

7.4.3. Requerimientos de Hardware . . . 162

7.5. Principales funcionalidades . . . 162

7.5.1. Conversión y tratamiento de colecciones de datos . . . 163

7.5.2. Entrenamiento y prueba de modelos . . . 167

7.5.3. Comparación y análisis de modelos . . . 169

7.6. Un caso de estudio 172
7.7. Conclusiones 175

PARTE V. Conclusiones y Trabajos futuros. 176

8. Conclusiones y Trabajos futuros 178

8.1. Conclusiones Generales. 178
8.2. Trabajos futuros. 182

Apéndice A. Especificaciones sobre la producción científica del L2R 185

Apéndice B. Rendimientos obtenidos por RankPSO en comparación con métodos publicados. 189

Apéndice C. Resultados del estudio estadístico en colecciones estándares de prueba. 192

Apéndice D. Resultados del estudio estadístico en la colección MSLR-WEB30K. 197

Bibliografía 213

Índice de figuras

1-1. Relación de objetivos por capítulos. 6

2-1. Representación del problema del L2R. 12
2-2. Esquema general del Aprendizaje de la Ordenación para la RI. Imagen mejorada, procedente de Tie-Yan Liu [1]. 14
2-3. Taxonomía de los problemas del L2R. Imagen procedente de Hang Li [2]. . 14
2-4. Línea de tiempo y tendencia categórica del aprendizaje de la ordenación. . 28

3-1. Convergencia de las disciplinas métricas. 42
3-2. Evolución del L2R. 46
3-3. Temáticas más exploradas en L2R. 47
3-4. Áreas del conocimiento en las que más se aplica el L2R. 48
3-5. Autores de mayor productividad científica en L2R. 49
3-6. Autores más referenciados en los últimos 10 años. 50
3-7. Países con mayor productividad científica en L2R. 52
3-8. Temáticas en auge y estables en el año 2009. 53
3-9. Temáticas en auge y estables en el año 2013. 54
3-10. Esquema de temáticas incipientes en L2R. 57
3-11. Esquema simbólico que representa la tendencia a la obsolescencia en el tratamiento de algunas temáticas en L2R. 58
3-12. Temáticas más tratadas en la Conferencia Internacional Anual del SIGIR. . 62
3-13. Temáticas más tratadas en las revistas más relevantes. 63
3-14. Red de colaboración autoral en L2R. 64

4-1. Movimiento que experimenta una partícula en PSO - La nueva posición ubica a la partícula en una zona intermedia entre las memorias personal y global. 71
4-2. Posición de las partículas al finalizar la primera iteración. 77

4-3. Posición de la partícula x_1 al finalizar la segunda iteración. 79
4-4. Clústeres clasificados sobre el espacio PCA para los dos primeros PCs. . . . 84
4-5. Representación de la calidad de las particiones, considerando el índice de Silhouette. 85

5-1. Esquema gráfico que representa el movimiento del pescador en un espacio unidimensional. 101
5-2. Representación gráfica de las cinco funciones de prueba de De Jong. 108
5-3. Resultados obtenidos en las funciones de De Jong, variando la cantidad de puntos de captura. 109

6-1. Primer lanzamiento de la malla del pescador. 124
6-2. Segundo lanzamiento de la malla del pescador. 125
6-3. Estadísticos de contraste obtenidos para ambas colecciones de datos. 144
6-4. Resumen de las comparaciones múltiples por parejas de los algoritmos en MSLR-WEB10K. 145

7-1. Diagrama de clases del *framework* incluido en L2RLab - Algunos métodos, atributos y parámetros son excluidos para una mejor comprensión. 160
7-2. Interfaz principal de la aplicación L2RLab. 161
7-3. Fichero de configuración en L2RLab. Cada método implementado está asociado con un fichero similar, el cual permite establecer valores a los atributos de la clase sin tener que modificar el código fuente. 162
7-4. Pantalla del módulo n^o.1 que muestra información específica de una colección de datos. 165
7-5. Pantalla del módulo n^o.1 que muestra las opciones posibles para modificar la información de una colección de datos. 166
7-6. Pantalla del módulo n^o.1 que muestra las opciones para generar múltiples ficheros a partir del fichero principal de una colección de datos. 167
7-7. Pantalla del módulo n^o.2 que permite realizar entrenamientos para evaluar el comportamiento de los modelos de L2R. 168

7-8. Pantalla del módulo n^o.3 que posibilita seleccionar la medida de evaluación, el conjunto de datos, y los modelos de aprendizaje que serán comparados. . 169
7-9. Pantalla del módulo n^o.3 que visualiza la precisión en el *ranking* alcanzada por múltiples modelos de ordenación a nivel de consulta. 170
7-10. Pantalla del módulo n^o.3 que visualiza para n (=20) iteraciones el comportamiento de las curvas de aprendizaje de múltiples modelos de L2R. 171
7-11. Pantalla del módulo n^o.3 que permite realizar pruebas no paramétricas, considerando el rendimiento de los métodos de L2R a nivel de consulta. . . 172

A-4. Redes de colaboración entre autores de L2R. 188
D-1. Resumen de las comparaciones múltiples por parejas de los algoritmos en MSLR-WEB30K. 197

Índice de cuadros

3-1. Temáticas más tratadas por los 50 autores de mayor impacto. 60

4-1. Ejemplo sencillo de conjunto de entrenamiento para el L2R. 76

4-2. Rendimientos obtenidos por los diferentes algoritmos de L2R sobre los *datasets* de la colección .Gov, considerando MAP como medida de evaluación. 89

4-3. Rendimiento de algoritmos estudiados, considerando NDCG@1, NDCG@5 y NDCG@10. 91

4-4. Coeficiente de variación (%) alcanzado por *RankPSO-Div-Red* en todos los *datasets* y considerando MAP, P@n y NDCG@n como medidas de evaluación. 92

4-5. Tiempo de preprocesamiento (en segundos) empleado en cada *dataset* estudiado. 96

4-6. Tiempo total de procesamiento (en segundos) empleado en cada *dataset* para cada variante. 97

5-1. Equivalencias del pescador entre el modelo físico y el modelo de optimización. 102

5-2. Ecuaciones y atributos de las cinco funciones de prueba de De Jong. 107

5-3. Resultados promedios obtenidos para diferentes números de iteraciones y lanzamientos. 111

5-4. Parámetros recomendados para FSP según su evaluación ante las funciones de prueba de De Jong. 112

5-5. Funciones de referencia para evaluar métodos heurísticos. 114

5-6. Rendimientos alcanzados por los algoritmos frente a cada función de prueba. 117

6-1. Ejemplo sencillo de conjunto de entrenamiento para el L2R. 123

6-2. Rendimientos alcanzados por algoritmos de L2R sobre los *datasets* MSLR-WEB10K (a) y MSLR-WEB30K (b), considerando MAP como medida de evaluación. 141

6-3. Rendimientos alcanzados en MSLR-WEB10K (a) y MSLR-WEB30K (b), considerando NDCG en las 10 primeras posiciones del *ranking*. 142
6-4. Rendimientos alcanzados en MSLR-WEB10K (a) y MSLR-WEB30K (b), considerando la medida *Precision at n* en las 10 primeras posiciones del *ranking*. 142
6-5. Tiempo en horas empleado para el preprocesamiento de cada conjunto de entrenamiento. 147
6-6. Tiempo de procesamiento (horas) para construir una función de *ranking* a partir de cada conjunto de entrenamiento. 147
6-7. Número de consultas evaluadas en cada conjunto de datos según el esquema utilizado. 148

7-1. Resumen de las principales colecciones de datos utilizadas en el área del L2R. 163
7-2. Rendimientos obtenidos por el modelo tratado sobre la colección de OHSUMED. 174

A-1. Tabulación de la producción científica del L2R por área del conocimiento. . 185
A-2. Tabulación de los autores con mayor productividad científica en L2R. . . . 186
A-3. Tabulación de la producción científica por países. 187
B-1. Rendimiento de algoritmos estudiados, considerando P@1 hasta P@10 en TD2004. 189
B-2. Rendimiento de algoritmos estudiados, considerando P@1 hasta P@10 en TD2003. 190
B-3. Rendimiento de algoritmos estudiados, considerando P@1 hasta P@10 en NP2004. 190
B-4. Rendimiento de algoritmos estudiados, considerando P@1 hasta P@10 en NP2003. 190
B-5. Rendimiento de algoritmos estudiados, considerando P@1 hasta P@10 en HP2004. 191
B-6. Rendimiento de algoritmos estudiados, considerando P@1 hasta P@10 en HP2003. 191

C-1. Estadísticos obtenidos por la prueba de Friedman en HP2003. 192
C-2. Rangos en HP2003. 192
C-3. Estadísticos obtenidos por la prueba de Friedman en HP2004. 192
C-4. Rangos en HP2004. 193
C-5. Estadísticos obtenidos por la prueba de Friedman en NP2003. 193
C-6. Rangos en NP2003. 193
C-7. Estadísticos obtenidos por la prueba de Friedman en NP2004. 194
C-8. Rangos en NP2004. 194
C-9. Estadísticos obtenidos por la prueba de Friedman en TD2003. 194
C-10.Estadísticos obtenidos por la prueba de Friedman en TD2004. 194
C-11.Rangos en TD2003. 195
C-12.Rangos en TD2004. 195

Índice de algoritmos

4.1. Algoritmo PSO 72
4.2. Algoritmo RankPSO 75

5.1. Algoritmo FSP 105

6.1. Algoritmo RankFSP 127

Notación

Q	...	Conjunto de consultas de una colección.
q_i	...	Denota la i-ésima consulta.
t	...	Término relativo a una determinada consulta.
$h(q_i)$	...	Cantidad de términos para la i-ésima consulta.
D	...	Conjunto de documentos de una colección.
d_i	...	Lista de documentos recuperados para la i-ésima consulta.
d_{ij}	...	Indica el j-ésimo documento en d_i.
Y	...	Definición n^o.1: Conjunto de juicios de relevancia.
y_i	...	Lista de etiquetas (juicios) de relevancia asociadas a d_i.
y_{ij}	...	Definición n^o.1: Denota la etiqueta de relevancia del documento d_{ij}.
$n(q_i)$	...	Cantidad de elementos de las listas d_i y y_i para la consulta q_i.
$\phi(q_i, d_{ij})$	...	Vector de rasgos asociado a un par consulta-documento.
S	...	Conjunto de entrenamiento.
$f(q_i, d_{ij})$	...	Función de *ranking* a nivel de documento.
w	...	Denota el vector de pesos.
π_i	...	Predicción de ordenación realizada por la función de *ranking* sobre d_i en términos de la consulta q_i.
$E(\pi_i, y_i)$	...	Función general para representar una medida de evaluación de RI.
π_i^*	...	Predicción ideal de la función de *ranking* para la consulta q_i.
Π_i^*	...	Conjunto de posibles predicciones ideales para la consulta q_i.
$R(f)$	...	Función de pérdida básica.
X	...	Conjunto de vectores de posición.
x_i	...	i-ésimo vector de posición.
v_i	...	i-ésimo vector de velocidad.
p_i	...	Memoria de la mejor solución local del elemento i.
g_{best}	...	Memoria de la mejor solución global.
n_1 y n_2	...	Vectores n-dimensionales formados por números aleatorios en el rango [0,1].
c_1	...	Constante de aceleración que influye en el movimiento de cada partícula i atrayéndola hacia su posición p_i.
c_2	...	Constante de aceleración que influye en el movimiento de las partículas atrayéndolas hacia la posición g_{best}.
V_{max}	...	Velocidad máxima que puede alcanzar una partícula en el espacio de solución.

w	...	Peso inercial.
T	...	Número de iteraciones.
P_1	...	Punto de referencia inicial que denota una determinada iteración.
P_2	...	Amplitud o rango de iteraciones a partir de la iteración P_1.
th	...	Umbral de cambio.
σ	...	Número de dimensiones del espacio de búsqueda.
τ	...	Indicador de la variabilidad o mejora que tienen las partículas en P_2.
CV	...	Coeficiente de variación.
N	...	Número de puntos de captura.
L	...	Número de lanzamientos de la red de pesca o malla en un punto de captura.
M	...	Número de vectores de posición de la malla.
Y	...	Conjunto de vectores de posición de la malla (Definición n^o.2).
y_{ij}	...	Denota el j-ésimo vector de posición de la malla (Definición n^o.2). en el i-ésimo punto de captura.
c	...	Coeficiente de amplitud.
A_j	...	Vector n-dimensional compuesto por números aleatorios en el rango [-c,c].
DP	...	Rango de iteraciones en el que se mide el desarrollo de la pesca.
δ	...	Umbral (en %) que denota la cantidad de consultas de una colección de datos cubiertas positivamente por un subconjunto de rasgos.
F	...	Constante que representa la energía potencial gravitatoria.
G	...	Constante que representa la gravitación universal.
λ	...	Fuerza de la explosión.
α	...	Tasa de disminución en el tiempo de la fuerza de explosión.

PARTE I

Introducción General

Capítulo 1

Introducción y motivación

El *ranking* constituye el problema clave en muchas aplicaciones de Recuperación de Información[1](RI). Estas incluyen recuperación de documentos[2], filtraje colaborativo[3], extracción de términos claves[4], metabuscadores[5], búsqueda personalizada[6], búsqueda de respuestas[7], mercadotecnia en Internet[8] y traducción automática[9].

Básicamente, existen dos tipos de problemas de ordenación: la creación de ordenación[10](ó simplemente ordenación) y la agregación de ordenación[11]. Por su parte, la creación de ordenación está dirigida a crear una lista ordenada de objetos usando los rasgos de tales objetos, mientras que la agregación de ordenación se basa en crear una lista ordenada de objetos a partir de múltiples listas ordenadas de objetos. [2]

Estos problemas de ordenación han sido perfilados y tratados, dentro de una nueva área de investigación que emerge como intersección del ML y la RI, el Aprendizaje de la Ordenación, denominado en inglés *Learning to rank* (L2R).

El interés por el L2R es relativamente reciente, desde hace alrededor de 15 años. A pesar de ello, se han llevado a cabo un gran número de estudios sobre L2R [3–7] y sus aplicaciones a la RI [8–10].

En general, todos aquellos métodos que usan tecnologías de ML para resolver problemas de ordenación pueden ser llamados métodos de aprendizaje de la ordenación. Estos nuevos métodos de L2R se basan en el aprendizaje supervisado y persiguen sobre cualquier escenario, crear de forma automática un modelo de ordenación usando datos de entrenamiento, técnicas de aprendizaje automático y medidas de evaluación utilizadas en la RI.

[1]Traducción literal del término inglés "Information Retrieval (IR)".
[2]Traducción literal del término inglés "Document Retrieval".
[3]Traducción literal del término inglés "Collaborative Filtering".
[4]Traducción literal del término inglés "Key Term Extraction".
[5]Traducción literal del término inglés "Metasearch engine".
[6]Traducción literal del término inglés "Personalized search".
[7]Traducción literal del término inglés "question answering".
[8]Traducción literal del término inglés "online advertising" o "Internet advertising".
[9]Traducción literal del término inglés "machine translation".
[10]Traducción literal del término inglés "ranking creation".
[11]Traducción literal del término inglés "ranking aggregation".

Según la manera en que modelan y tratan el problema de L2R, los métodos existentes se encuentran enmarcados en tres categorías fundamentales: la primera es el Enfoque por Puntos (*Pointwise Approach*), en la cual se transforma la ordenación en una clasificación (por ejemplo, [11], McRank [12]), regresión ([13]) o regresión ordinal (Pranking [14] y [15]) sobre documentos simples. La segunda categoría es considerada como el Enfoque por Pares (*Pairwise Approach*) (por ejemplo, RankingSVM [16], RankBoost [17], RankNet [18], FRank [19] y MHR [20]), en la cual de manera general se transforma o formaliza el problema de la ordenación en una clasificación binaria sobre pares de documentos. La tercera es referenciada como Enfoque por Listas (*Listwise Approach*) (por ejemplo, AdaRank [9], SVMmap [21], SoftRank [22], LambdaRank [23] y [24]), en la cual se minimiza directamente una función de pérdida definida sobre una lista de documentos.

Sin embargo, a pesar de los aportes y contribuciones de cada uno de estos métodos en el área del L2R. Todavía las soluciones sólo garantizan una aproximación local, los métodos que trabajan sobre límites o cotas superiores [5], muchas veces presentan cotas muy holgadas, relajando en demasía la solución. Otros métodos llevan a cabo entrenamientos sesgados debido a una función de pérdida global dominada por aquellas consultas con un gran número de documentos y en otros casos, la función de pérdida pasa por desapercibido la posición que van ocupando los documento en la lista ordenada. También el proceso de transformar las múltiples categorías de relevancia a preferencias a pares entre documentos puede propiciar pérdida de información e introducir instancias ruidosas. Por otro lado, debido a que las medidas de evaluación de RI son funciones complejas, discontinuas y no diferenciables, la tarea de optimizar directamente cualquiera de estas métricas es muy difícil y conlleva a problemas de complejidad algorítmica y a soluciones locales.

Además, resulta conveniente concebir métodos de L2R encaminados a alcanzar entrenamientos escalabes y eficientes, que fuesen capaces de incorporar algún tipo de estrategia para evadir la convergencia temprana hacia mínimos locales, que pudiesen reducir y mejorar los tiempos de entrenamiento frente a las grandes colecciones de datos. También, sería ventajoso que estos métodos aprendiesen a utilizar y combinar mejor los rasgos según las diversas condiciones y contextos en las que se presentan las colecciones de datos.

Tales peculiaridades existentes, limitan las potencialidades de rendimiento de las propues-

tas actuales, dejando abierto un campo amplio a la investigación. Precisamente, en este sentido, hemos enmarcado el trabajo desarrollado en nuevos modelos de L2R basados en técnicas de Soft Computing; con las premisas fundamentales de solucionar algunos problemas anteriores, diseñar y validar métodos dirigidos a una mejor escalabilidad y estabilidad de rendimiento y, consecuentemente, incrementar las precisiones alcanzadas en la ordenación por los métodos referenciados en la literatura.

Objetivos

El objetivo principal de esta investigación es el desarrollo de modelos de L2R que incrementen la calidad en el *ranking* de documentos generados. Este objetivo genérico se puede descomponer en los siguientes específicos:

1. Obtener una comprensión conceptual del "*Learning to rank*" aplicado a la RI, del planteamiento general del problema y los enfoques de partida que se han establecido en esta área.

2. Evaluar el comportamiento de la producción científica sobre L2R, revelando las principales tendencias y caracterizando el estado actual de las líneas temáticas en esta área.

3. Proponer nuevos algoritmos de L2R basados en técnicas de soft computing o en heurísticas especialmente diseñadas, que sean capaces de optimizar directamente cualquier medida de evaluación utilizada en la RI y permitan mejorar la precisión en el ranking.

4. Determinar el rendimiento que se puede alcanzar con los modelos diseñados a través de su evaluación en colecciones documentales estándar. Comparando y analizando estadísticamente tales rendimientos con respecto a los publicados por los principales métodos referenciados en la literatura.

5. Mejorar la calidad de actuación, el poder de generalización y el tiempo de respuesta de los modelos de ordenación, mediante el diseño o adaptación de procedimientos para reducir la dimensionalidad de los conjuntos de datos utilizados.

6. Elaborar una herramienta software para asistir experimentalmente la labor de los investigadores durante la tarea del L2R.

Descripción de la memoria por capítulos

Con el propósito de estructurar correctamente la exposición de los resultados de esta investigación, la presente memoria queda organizada de la siguiente manera:

Capítulo 2. En este capítulo introduciremos formalmente el problema del L2R para la RI. Describiremos las principales categorías y modelos en general, siguiendo su devenir histórico y lógico. Se abordarán además, las diversas vertientes enfocadas en optimizar directamente cualquier medida de evaluación en RI. Finalmente, se hará alusión a las medidas de evaluación utilizadas en este campo y a las colecciones estándares establecidas como referencia por la comunidad científica. Todo lo cual nos facilitará la comprensión del tema que se investiga.

Capítulo 3. Este capítulo presenta un estudio bibliométrico desarrollado a partir de registros bibliográficos de la Base de Datos Scopus[12] para evaluar la producción científica del L2R durante el período del año 2000 al 2013. Se describen los principales conceptos asociados a los estudios métricos de la información relacionados con las ciencias de la información, y se plantea el procedimiento para llevar a cabo el estudio métrico. Se detallan y analizan los resultados obtenidos en dos grandes grupos de indicadores los unidimensionales y los multidimensionales, con el apoyo de gráficos ilustrativos y mapas topológicos. Todo lo cual nos permitió conocer cuantitativa y cualitativamente el comportamiento y las tendencias de la actividad científica y tecnológica sobre L2R, así como la estructura y dinámica social de los investigadores implicados.

Capítulo 4. En este capítulo se propone un nuevo método de L2R basado en la Optimización con Enjambre de Partículas (PSO, por sus siglas en inglés). Este algoritmo, que llamaremos RankPSO, es capaz de optimizar directamente cualquier medida de evaluación de RI e implementa una estrategia para evadir la convergencia temprana a soluciones locales. En la descripción de este método también se introduce un procedimiento de reducción de dimensionalidad combinando Análisis de Componentes Principales (PCA, por sus siglas en inglés) y Análisis de Clúster. Se detalla también una fase experimental utilizando los

[12]Disponible en http://www.scopus.com/home.url

conjuntos de datos de ".Gov" para evaluar el rendimiento de RankPSO y su efectividad en comparación con métodos referenciados en la literatura. Además, se describe un análisis estadístico llevado a cabo sobre tales resultados, y finalmente se expone un análisis de la influencia del procedimiento de reducción de dimensionalidad con respecto al coste computacional.

Capítulo 5. En este capítulo se propone una novedosa metaheurística para problemas de optimización global, que denominaremos Procedimiento de Búsqueda del Pescador (FSP, por sus siglas en inglés). Las bases de la idea de este modelo son explicadas en los primeros apartados, donde se describe también una fase de parametrización en la que se analiza el comportamiento de los parámetros de este algoritmo considerando las cinco funciones de prueba de De Jong. Luego, se detalla toda una etapa de prueba donde se evalúan las funcionalidades de esta metaheurística mediante un conjunto de funciones utilizadas para validar métodos heurísticos. En esta fase de prueba se exponen además, las comparaciones realizadas con respecto a otros métodos heurísticos referenciados en la literatura en relación a la capacidad de convergencia a valores mínimos globales y el coste de tiempo de procesamiento.

Capítulo 6. En este capítulo se describe una variante mejorada y adaptada del Procedimiento de Búsqueda del Pescador para tratar el problema del L2R. Este nuevo método de L2R, denominado RankFSP, es capaz de construir una función de ordenación optimizando directamente cualquier medida de evaluación según un determinado conjunto de entrenamiento. Se explica además, como se incorporan en RankFSP dos nuevas estrategias para mejorar la búsqueda y evadir mínimos locales o puntos de búsqueda estancados. Se justifica y se enuncia también un método simple de selección de rasgos. Posteriormente, se detalla todo un estudio experimental donde se evalúo el rendimiento de RankFSP y su efectividad en comparación con otros métodos referenciados en la literatura. Los resultados obtenidos fueron analizados estadísticamente. Finalmente, se presenta un análisis del coste computacional en relación al tiempo de procesamiento y a la cantidad de consultas evaluadas.

Capítulo 7. En este capítulo presentamos la herramienta *Learning-to-Rank Laboratory* (L2RLab), que tiene como objetivo principal asistir y facilitar la labor del investigador

durante las fases experimentales necesarias para desarrollar y validar nuevos modelos de L2R. En los primeros apartados se explican las concepciones básicas del diseño de experimentos, y luego se detallan los aspectos técnicos de L2RLab. Luego, se describen las funcionalidades de cada uno de los tres módulos de la parte visual. Para concluir, se debate un caso de estudio donde se muestra cómo incluir un nuevo algoritmo a L2RLab.

Capítulo 8. Finalmente, en este capítulo exponemos las conclusiones de todo el proceso investigativo, y proponemos varias líneas interesantes que pueden ser abordadas en trabajos futuros.

A continuación se presenta un gráfico relacional donde se muestra en qué capítulos de esta memoria serán tratados cada uno de los objetivos propuestos.

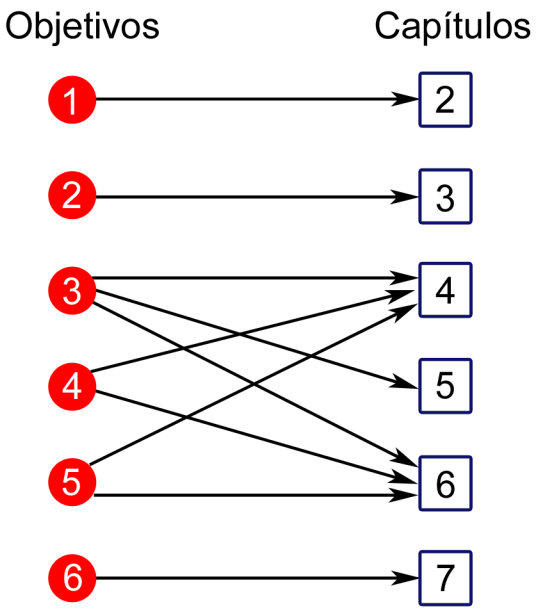


Figura 1-1: Relación de objetivos por capítulos.

Capítulo 2

Introducción al Aprendizaje de la Ordenación en Recuperación de Información

"La verdad científica es roca firme,
en que estamos autorizados a cimentar
los más sólidos edificios".
— Carlos J. Finlay

2.1. Introducción

El presente capítulo aborda de forma panorámica y conceptual las premisas fundamentales del marco teórico relativo al Aprendizaje de la Ordenación para la Recuperación de Información. Partiendo de la descripción y formulación del problema en cuestión, hasta la definición y desglose de las principales categorías y modelos en general. Abordando en particular, las propuestas encaminadas a optimizar directamente cualquier medida de rendimiento. Finalizaremos, enunciando las colecciones estándar de prueba y las medidas de evaluación más utilizadas y establecidas como referencia por la comunidad científica que investiga en esta área.

2.2. Modelos convencionales de RI

Los modelos convencionales de RI representan un documento como un conjunto de palabras claves representativas, las cuales se traducen en términos que instancian rasgos o características de los documentos, por los cuales son indexados. Estos modelos tradicionales definen una función de recuperación para asociar un grado de relevancia a un documento y una consulta dada, permitiendo obtener un *ranking* de documentos. A partir de entonces y de una manera no supervisada, los parámetros que sustentan dichas funciones son ajustados empíricamente.

Los principales modelos convencionales de RI comprendidos en el período de los años 1970 hasta la actualidad, y que marcan un salto fundamental en este campo, se encuentran enmarcados en dos categorías principales:

1. **Modelos dependientes de la consulta.** El primer modelo clásico de RI fue el Modelo Booleano (*Boolean Model*) [25]. Este modelo está basado en la lógica booleana y la teoría clásica de conjuntos, donde los documentos a buscar y la consulta del usuario, son concebidos como un conjunto de términos (palabras o frases). La recuperación está basada en cuando los documentos contienen o no los términos de la consulta. También fueron muy acogidos los modelos algebraicos como el Modelo de Espacio Vectorial (*Vector Space Model*) (VSM) [26] [27] [28] y el Modelo de Indexación Semántica Latente (*Latent Semantic Indexing*) (LSI) [29], donde este último constituye una variación del primero. En estos dos modelos los documentos y las consultas se representan de manera formal como vectores (índices de términos), en el VSM estos vectores son mapeados en un espacio de términos indexados y en el LSI en un espacio dimensional reducido el cual está asociado con conceptos. Para ambos modelos la relevancia de un documento frente a una consulta es calculada a través de una función de similitud que está dada por el coseno del ángulo entre el vector del documento y el de la consulta. Por otra parte, los Modelos Probabilísticos (*Probabilistic Models*) [30] [31] [32] tratan el proceso de recuperación de documentos como una inferencia probabilística. Las similitudes son calculadas como las probabilidades de que un documento sea relevante dada una consulta. Otro de los modelos notables fueron los Modelos del Lenguaje para la RI (*Languaje Models for Information Retrieval, LMIR*) [33] [34] [35] [36] [37]. En este caso, se construye o asocia por cada documento de la colección un determinado modelo del lenguaje. Cuando los documentos son recuperados para una consulta, estos son ordenados sobre la base de la probabilidad de que el modelo del lenguaje de cada documento pudiese generar los términos de dicha consulta. Esto se interpreta como la probabilidad de que un documento sea relevante dada una consulta.

2. **Modelos independientes de la consulta.** Esta categoría de modelos clasifican sus documentos en base a su propia importancia o relevancia y no en relación a una

determinada consulta. Uno de los modelos más importantes es PageRank [38]. Es el algoritmo utilizado por *Google* para asignar de forma numérica la relevancia de los documentos (o páginas web) indexados por su motor de búsqueda. El PageRank de una página se define recursivamente y depende de la calidad de la página y el PageRank de todas las páginas que la enlazan. Una página que está enlazada por muchas páginas con un PageRank alto consigue también un PageRank alto. Si por el contrario nadie la enlaza, su valor de PageRank es muy bajo. El valor numérico que representa el PageRank de una página va desde 0 a 10. Diez es el máximo PageRank posible y son muy pocas las páginas que tienen esta puntuación, 1 es la calificación mínima que recibe una página normal, y cero significa una página penalizada o que aún no ha recibido una calificación de PageRank. Los detalles exactos de esta escala son desconocidos. Posteriormente, siguiendo la idea base de PageRank de considerar la estructura de enlaces para generar una medida de la calidad de una página, fue desarrollado el algoritmo TrustRank [39]. Este procedimiento parte de un proceso manual donde se seleccionan buenas páginas o páginas de confianza. Estas páginas van a constituir las fuentes de confianza. Esta medida de confianza se propaga de la misma manera como lo hace el PageRank, es decir, sólo puede ser transferida a otra página mediante un enlace hacia ella. En este método, se seleccionan además, las fuentes de spam. Esta medida negativa (como un PageRank inverso) se propaga hacia atrás y es una medida de las malas páginas (spam). TrustRank considera ambas medidas para evaluar la calidad de los sitios web. Como ejemplo final podemos mencionar al método BrowseRank [40]. Este algoritmo calcula la importancia de una página mediante el uso de un "grafo de navegación de usuario" creado a partir de datos (históricos) que registran el comportamiento de los usuarios en la web. En este grafo, los vértices representan páginas y la dirección de las aristas representan transiciones entre las páginas. En este esquema también se incluyen los tiempos de estancias que tuvieron los usuarios en cada una de las páginas. Sobre este grafo, BrowseRank usa como modelo el Proceso de Markov de tiempo continuo [41], donde el cálculo de la distribución de probabilidad estacionaria de dicho proceso constituirá la importancia de cada página.

A pesar de que estos modelos han propiciado un desarrollo importante dentro del campo de la RI, tales métodos convencionales se enfrentan a las siguientes limitantes:

- Ajustar los parámetros manualmente es usualmente difícil, especialmente cuando son muchos los parámetros y las medidas de evaluación son *non-smooth*[1].
- Ajustar estos parámetros manualmente algunas veces nos puede conducir a un sobreajuste.
- No resulta trivial combinar o buscar oportunidades híbridas de todos los modelos propuestos en la literatura para obtener siempre uno más efectivo.

Solventar tales problemas requiere del uso de técnicas y estrategias de aprendizaje automático como herramientas efectivas para:

- Ajustar los parámetros de forma automática.
- Combinar múltiples evidencias.
- Evitar el sobreajuste (por medio de la regularización, etc.).

2.3. Descripción del problema del L2R

El L2R es un área creciente de investigación que ha emergido como intersección necesaria entre la RI y el ML. En un sentido amplio, el L2R puede entenderse como cualquier técnica de ML utilizada en una tarea de *ranking*. Sin embargo, en un sentido estricto, el L2R se refiere específicamente a las técnicas de ML utilizadas para construir modelos para la creación de *ranking*[2] y modelos para la agregación de *ranking*[3].
En la investigación que se presenta en esta memoria documental, haremos alusión a esta segunda definición, donde particularmente, nos enfocaremos en la construcción de modelos para la creación de *ranking*.

[1]Término inglés que se refiere en dicho contexto a una función discontinua y no diferenciable.
[2]Traducción literal del término inglés "ranking creation".
[3]Traducción literal del término inglés "ranking aggregation".

A partir de esta premisa contextual, podemos decir que todos aquellos métodos que usan tecnologías de ML para resolver problemas de creación de *ranking* pueden ser llamados, métodos de L2R. Estos nuevos métodos de L2R se basan en el aprendizaje supervisado y persiguen sobre cualquier escenario crear, de forma automática, un modelo de *ranking* usando datos de entrenamiento, técnicas de ML y medidas de evaluación utilizadas en la RI [2].

Para describir el problema del L2R para la RI de la manera más sencilla e ilustrativa, consideremos el siguiente ejemplo representado a través del esquema de cinco pasos de la Figura 2-1.

En el primer paso, tenemos un usuario deseoso de información que formula una consulta a un determinado motor de búsqueda (por ejemplo, Google, Yahoo o Bing). Los documentos recuperados para esta consulta fueron el d_1, d_2 y d_3. En un segundo paso, se le aplica a cada documento un mecanismo de extracción de rasgos considerando los términos (palabras) de la consulta. Este procedimiento calcula desde rasgos sencillos como la frecuencia de aparición de los términos de la consulta en el cuerpo del documento, hasta rasgos más complejos como el BM25 [42] y el PageRank [38]. Luego, para el paso tercero, tenemos un especialista humano que etiqueta cada documento según los siguientes grados de relevancia con respecto a la consulta: totalmente relevante "2", parcialmente relevante "1" y no relevante "0". A estas etiquetas se les denomina *ground truth*[4]. El d_1 fue etiquetado con "1", el d_2 con "2" y el d_3 con "0". La finalidad de estos pasos dos y tres, es construir por cada documento, un vector de rasgos representativo que esté compuesto por la etiqueta de relevancia y a continuación un listado de cada rasgo extraído con su valor correspondiente. Estos tres vectores van a constituir el conjunto de entrenamiento que será utilizado durante el proceso de aprendizaje. Por supuesto, para un caso real, estos pasos del uno al tres se repiten miles de veces y finalmente, se tiene un conjunto de datos que adiciona a sus vectores un identificador de la consulta, debido a que un mismo documento puede ser recuperado para más de una consulta.

[4]Término que representa la realidad tal y como es, en el caso de etiquetas *ground truth* hace alusión al valor real del documento clasificado según el juicio de especialistas.

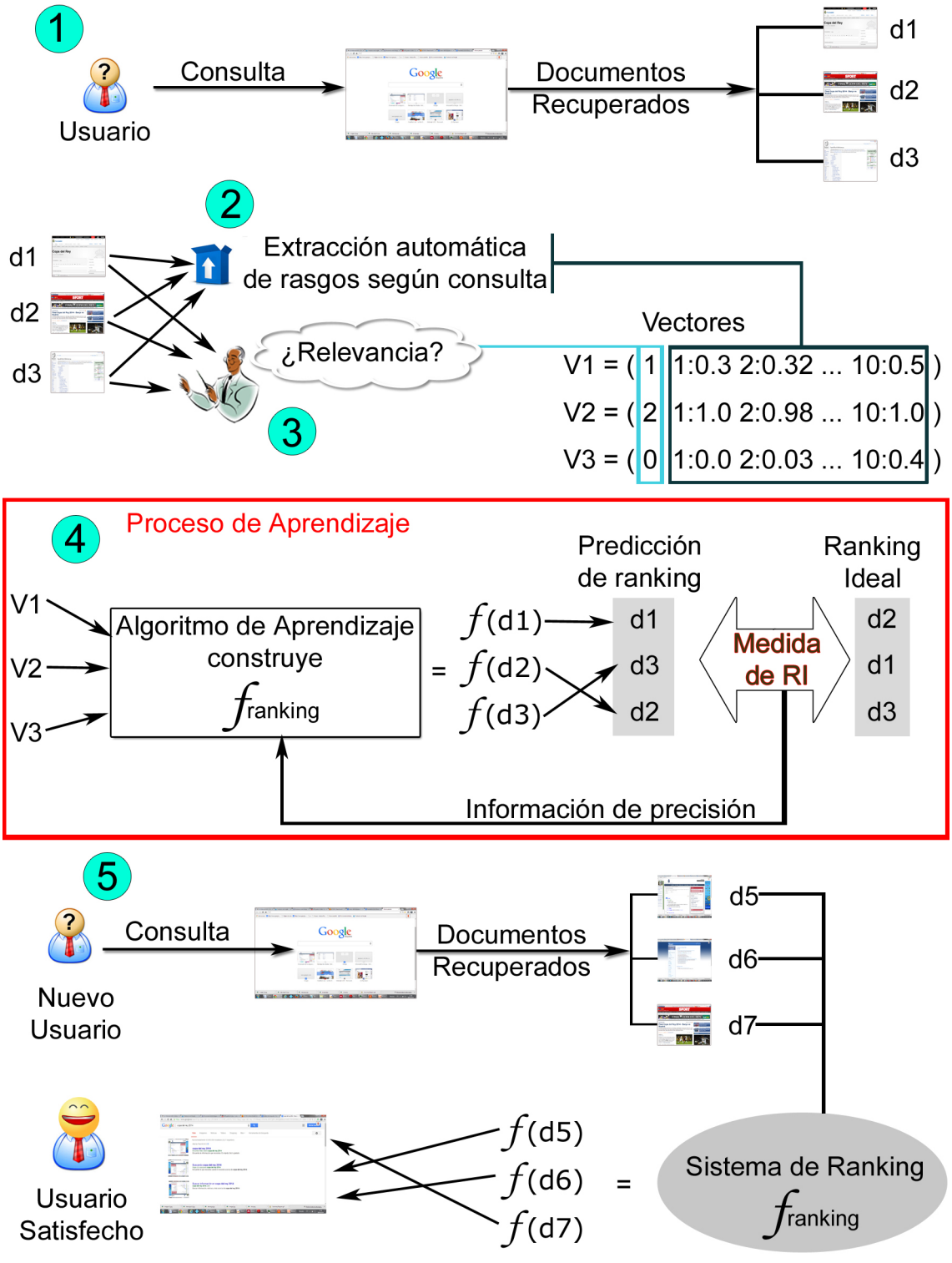


Figura 2-1: Representación del problema del L2R.

El paso cuatro (limitado por un rectángulo de bordes rojos) ilustra de manera general el proceso de aprendizaje. Tenemos un algoritmo de aprendizaje basado en ML que a partir del conjunto de entrenamiento va construyendo una función de *ranking*.

Durante cada ciclo o iteración esta función de *ranking* generalmente de la forma: $f : Q \times D \rightarrow \mathbb{R}$ [5] asigna una puntuación real a cada documento y a continuación ordena descendentemente tales documentos a partir de sus puntuaciones de relevancia. Entonces utilizando una medida de evaluación de RI se determina la correspondencia entre el *ranking* predicho por el modelo y el ranking *ideal* basado en las etiquetas *ground truth*. Para un análisis más detallado de estas métricas de RI utilizadas en L2R, consultar la Sección 2.7. Esta información de precisión en el *ranking* dada por la medida de evaluación, es utilizada por el modelo de aprendizaje para realizar sus ajustes pertinentes (por ejemplo, pesos, parámetros). A partir de entonces, el objetivo del algoritmo de aprendizaje es entrenar un modelo que optimice el *ranking* basado en una medida de evaluación y con respecto a los datos de entrenamiento. Este proceso de aprendizaje se repite hasta cumplir un criterio de parada o abarcar un cierto número de iteraciones.

Una vez ajustado el modelo de *ranking*, en el proceso de entrenamiento, se prosigue a la etapa de prueba (paso cinco), donde dada una nueva consulta y una lista de documentos recuperados (d_5, d_6 y d_7), el sistema de *ranking* tiene que ser capaz de retornar una lista de documentos en orden representativo de acuerdo a sus valores de relevancia, de tal forma que en la lista recuperada, los documentos relevantes sean ordenados en mejores (altas) posiciones con respecto a aquellos que son menos relevantes o irrelevantes. Idealmente, la función de *ranking* para esta etapa, logrará alcanzar resultados lo suficientemente buenos en cuanto a precisión y, por lo tanto, en cuanto a satisfacción del usuario.

En la Figura 2-2 se presenta un esquema general y sintetizado del L2R que abarca cada una de estas etapas explicadas.

A modo de conclusión, en la Figura 2-3 se muestra cómo está concebida la estructura problemática del L2R para la RI. Básicamente, existen dos tipos de problemas de ordenación: la creación de *ranking* (ó simplemente ordenación) y la agregación de *ranking*. Por

[5]Función aplicada a los documentos recuperados del conjunto $\boldsymbol{D}$ con respecto a la correspondiente consulta perteneciente al conjunto $\boldsymbol{Q}$

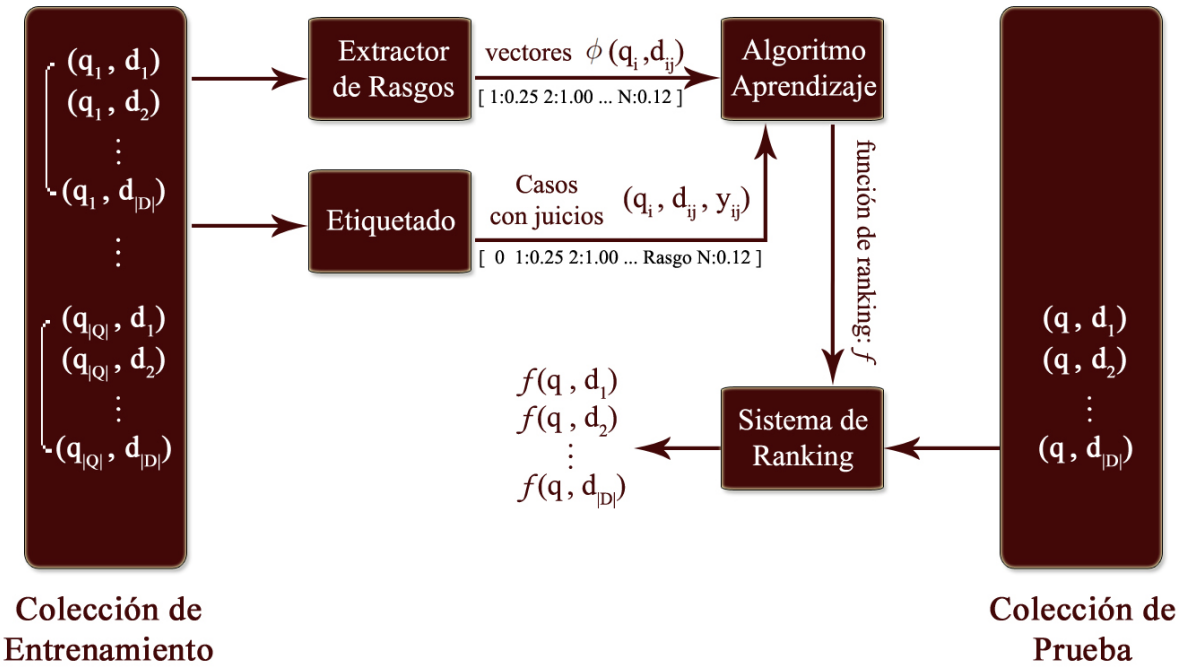


Figura 2-2: Esquema general del Aprendizaje de la Ordenación para la RI. Imagen mejorada, procedente de Tie-Yan Liu [1].

su parte, y como ya hemos visto, la creación de *ranking* está dirigida a crear una lista ordenada de objetos (documentos) usando los rasgos de tales objetos, mientras la agregación de *ranking* se basa en crear una lista ordenada de objetos a partir de múltiples listas ordenadas de objetos [2].

$$Learning\,to\,Rank \begin{cases} Ranking\,Creation \begin{cases} Supervised\,(ej., Ranking\,SVM) \\ Unsupervised\,(ej., BM25) \end{cases} \\ Ranking\,Aggregation \begin{cases} Supervised\,(ej., CRank) \\ Unsupervised\,(ej., Borda\,Count) \end{cases} \end{cases}$$

Figura 2-3: Taxonomía de los problemas del L2R. Imagen procedente de Hang Li [2].

Particularmente, y como ya hemos mencionado, la presente investigación está encaminada a tratar el problema de la creación de *ranking* siguiendo específicamente un enfoque supervisado.

2.4. Formulación general del L2R

Supongamos que $Q = \{q_1, q_2, \ldots, q_m\}$ es el conjunto de consultas en el entrenamiento, donde D constituye la colección documental recuperada para todas las consultas e $Y = \{r_1, r_2, \ldots, r_k\}$ es el conjunto de juicios de relevancia o etiquetas *ground truth*.

Un juicio de relevancia constituye una medida de cuan bueno es un documento recuperado con respecto a las necesidades de un usuario, reflejadas en la consulta formulada. Existe un orden total entre las etiquetas (valores) de relevancia, a decir, $r_k > r_{k-1} > \cdots > r_1$, donde $>$ denota el orden.

Cada consulta q_i está representada como una lista de términos $\{t_1, t_2, \ldots, t_{h(qi)}\}$, donde $h(q_i)$ denota la cantidad de términos para la i-ésima consulta. Toda consulta q_i está asociada con una lista de documentos recuperados $d_i = \{d_{i1}, d_{i2}, \ldots, d_{in(qi)}\}$ y una lista de etiquetas $y_i = \{y_{i1}, y_{i2}, \ldots, y_{in(qi)}\}$, donde el valor de $n(q_i)$ denota la cantidad de elementos de las listas $d_i \subseteq D$ y $y_i \subseteq Y$ para la consulta $q_i \in Q$; $d_{ij} \in d_i$ indica el j-ésimo documento en d_i, e $y_{ij} \in y_i$ denota la etiqueta del documento d_{ij}. Cada documento d_{ij} está representado como una lista de rasgos o características. Específicamente, un vector de rasgos $\phi(q_i, d_{ij})$ es creado para cada par consulta-documento (q_i, d_{ij}), donde $i = 1, 2, \ldots, m$ y $j = 1, 2, \ldots, n(q_i)$. Finalmente el conjunto de entrenamiento queda definido de la siguiente manera: $S = \{(q_i, d_i, y_i)\}_{i=1}^{m}$.

Considerando los patrones de la formulación descrita en [5], supongamos que π_i es la predicción de *ranking* realizada por el modelo de ordenación sobre d_i en términos de la consulta q_i. Se emplea Π_i para denotar el conjunto de todas las predicciones posibles sobre d_i, y se usa $\pi_{i(j)}$ para denotar la posición del elemento j, es decir, d_{ij}. El ordenamiento se concentra entonces en obtener una predicción $\pi_i \in \Pi_i$ para una consulta dada q_i y su lista de documentos asociados d_i, usando el modelo de ordenación.

El modelo de ordenación, que denotaremos como f, es una función lineal de valores reales que actúa a nivel de documento, y que en esencia, constituye una combinación lineal de los rasgos en un vector de rasgo $\phi(q_i, d_{ij})$. Esta función puede ser planteada de la siguiente manera:

$$f(q_i, d_{ij}) = w^T \phi(q_i, d_{ij}); \tag{2-1}$$

donde w denota el vector de pesos. Durante el proceso de creación de *ranking* a todos los documentos de la consulta q_i se les asigna una puntuación usando la función $f(q_i, d_{ij})$ y posteriormente tales documentos son ordenados en correspondencia con las puntuaciones que le fueron dadas. De tal forma se obtiene una predicción denotada como π_i.

En este trabajo se utiliza una función general $E(\pi_i, y_i) \in [0, 1]$ para representar las medidas de evaluación de RI (ver Sección 2.7). El primer argumento de E es la predicción π_i creada usando el modelo de ordenación sobre los documentos recuperados para una consulta q_i. El segundo argumento y_i es la lista de juicios, que determinan el orden ideal entre tales documentos para dicha consulta. El papel de E es, por tanto, medir la correspondencia entre π_i y y_i. La mayoría de las medidas de rendimiento retornan valores reales entre [0,1], mientras mayor sea el valor resultante de aplicar una de estas métricas mayor será la precisión alcanzada en el *ranking* por el modelo de L2R. En este sentido, la predicción ideal puede ser denotada como π_i^*.

Ahora, como puede existir más de una predicción ideal para una consulta, se denota Π_i^* como el conjunto de posibles predicciones ideales para la consulta q_i. Por tanto, $\pi_i^* \in \Pi_i^*$, y se tiene que $E(\pi_i^*, y_i) = 1$.

Idealmente, el objetivo de un modelo de ordenación es maximizar la precisión alcanzada sobre un conjunto de entrenamiento, en términos de una medida de evaluación de RI.

Para modelar este objetivo, podemos plantear la minimización de una función de pérdida básica definida en [5] como sigue:

$$R(f) = \sum_{i=1}^{m} \left(E(\pi_i^*, y_i) - E(\pi_i, y_i)\right) = \sum_{i=1}^{m} \left(1 - E(\pi_i, y_i)\right); \tag{2-2}$$

en la cual, cuanto más cercano a cero sea el valor resultante de esta expresión, más próxima estará la predicción π_i del modelo de *ranking* f para la consulta q_i al orden ideal.

2.5. Medidas de evaluación de RI utilizadas en L2R

Las principales medidas de evaluación empleadas en L2R para la RI, se dividen en dos categorías principales que son establecidas de acuerdo a los niveles de relevancia que contemplan. Con relevancia binaria tenemos las métricas: *Precision at n* (P@n) [43] y *Mean*

Average Precision (MAP) [43]. Por otra parte, con múltiples niveles de relevancia se sitúa la denominada *Normalized Discounted Cumulative Gain* (NDCG) [44].

Típicamente se emplean estas tres medidas de RI para evaluar la precisión en el ordenamiento. Estas pueden definirse como sigue:

Precision at n (P@n)

La precisión en n mide la relevancia de los documentos recuperados con respecto a una consulta, que se encuentran desde la primera posición del *ranking* hasta la posición n:

$$P@n = \frac{\#\, documentos\, relevantes\, en\, los\, primeros\, n\, resultados}{n}$$

Por ejemplo, si los 10 primeros documentos retornados a partir de una consulta son (relevante, irrelevante, irrelevante, relevante, relevante, relevante, irrelevante, irrelevante, relevante, relevante), entonces de P@1 a P@10 los valores serán (1, $\frac{1}{2}$, $\frac{1}{3}$, $\frac{2}{4}$, $\frac{3}{5}$, $\frac{4}{6}$, $\frac{4}{7}$, $\frac{4}{8}$, $\frac{5}{9}$, $\frac{6}{10}$), respectivamente [43].

Mean Average Precision (MAP)

Para una única consulta, la precisión media se define como el promedio de los valores de P@n para todos los documentos relevantes:

$$AP = \frac{\sum_{n=1}^{N} (P@n * rel\,(n))}{\#\, total\, de\, documentos\, relevantes\, para\, esta\, consulta}$$

donde N es el número de documentos recuperados, y *rel(n)* es una función binaria que retorna la relevancia del n-ésimo documento:

$$rel\,(n) = \begin{cases} 1, & si\, el\ n - esimo\ documento\, es\, relevante \\ 0, & en\, caso\, contrario \end{cases}$$

Por tanto, MAP es la media aritmética de la precisión media obtenida en cada una de las consultas evaluadas:

$$MAP = \frac{\sum_{q=1}^{Q} AP(Q)}{Q},$$

donde Q representa el total de consultas [43].

Normalized Discounted Cumulative Gain (NDCG)

NDCG es una medida de evaluación propuesta en el año 2002 que, a diferencia de P@n y MAP, puede manipular múltiples niveles de juicios de relevancia [44]. Esta métrica, mientras evalúa una lista de ordenamiento, sigue dos reglas fundamentales:

1. Los documentos altamente relevantes son más valiosos que los documentos marginalmente relevantes.

2. Un documento (de cualquier nivel de relevancia) de baja posición en el ordenamiento, tiene menos valor para el usuario, porque su probabilidad de ser examinado por dicho usuario es mucho menor.

De acuerdo a las reglas anteriores, para una consulta dada, el valor de NDCG para un documento situado en la posición i es calculado de la siguiente manera:

1. Calcular la ganancia de $2^{r(j)} - 1$ de cada documento j, donde *r(j)* es el valor de relevancia del j-ésimo documento en la lista ordenada;

2. Descontar la ganancia de cada documento j por su posición en el ranking, lo cual puede ser expresado como $\frac{(2^{r(j)}-1)}{\log(1+j)}$;

3. Acumular la ganancia descontada por documento j en posición i, que puede ser expresado por $\sum_{j=1}^{i} \frac{(2^{r(j)}-1)}{\log(1+j)}$;

4. Normalizar la ganancia acumulada descontada por documento j en posición i usando $n_i \sum_{j=1}^{i} \frac{(2^{r(j)}-1)}{\log(1+j)}$;

5. Finalmente podemos plantear que el valor NDCG de una lista ordenada en la posición i es calculado como sigue:

$$NDCG\,(i) = n_i \sum_{j=1}^{i} \frac{(2^{r(j)} - 1)}{\log\,(1 + j)},$$

y la constante de normalización n_i es seleccionada de tal forma que para la lista ideal se obtenga un valor NDCG igual a 1 [43].

Existen otras medidas de rendimiento dentro del ámbito de RI como ERR (Expected Reciprocal Rank) [45], RBP (Rank Biased Precision) [46], WTA (Winners Take All) [47], MRR (Mean Reciprocal Rank) [47], entre otras, que también han sido usadas pero de forma muy puntual tanto en trabajos teóricos como prácticos en el campo del L2R.

2.6. Principales categorías y modelos del L2R

Como se definió en la descripción del problema del L2R, los métodos de L2R se basan en el aprendizaje supervisado y persiguen sobre cualquier escenario, crear de forma automática, un modelo de ordenación usando datos de entrenamiento, técnicas de ML y medidas de evaluación utilizadas en RI.

Para categorizar tales propuestas algorítmicas en L2R, los autores del área se han apoyado en los cuatro componentes clave de un problema de ML:

- El **espacio de entrada**, que contiene los objetos de estudio generalmente representados mediante vectores de atributos.
- El **espacio de salida**, que contiene los objetivos (*targets*) con respecto a los objetos de entrada. En el caso de regresión este espacio de entrada suele ser $\mathbb{R}$, mientras que en clasificación se considera un conjunto discreto de clases o categorías.
- El **espacio de hipótesis**, que define el tipo de funciones que transforman un objeto de entrada en otro de salida.
- Una **función de pérdida o error**, que indica el grado de precisión de los resultados de las funciones del espacio de hipótesis respecto a salidas cuyo objetivo es de antemano conocido.

Considerando diversos criterios en los que están envueltos estos cuatro elementos, los métodos existentes de aprendizaje de la ordenación se encuentran enmarcados en tres categorías fundamentales: la primera es el Enfoque por Puntos (*Pointwise Approach*), la cual transforma la ordenación en una clasificación, regresión o regresión ordinal sobre documentos simples. La segunda categoría es considerada como Enfoque por Pares (*Pairwise*

Approach), en la cual de manera general se transforma o formaliza el problema de la ordenación en una clasificación binaria sobre pares de documentos. La tercera categoría hace referencia al Enfoque por Listas (*Listwise Approach*), en la cual se minimiza directamente una función de pérdida definida sobre una lista de documentos.

Cada enfoque tiene su propia definición de espacios de entrada y salida, usa diferentes hipótesis y emplea distintas funciones de pérdida. El espacio de salida se usa para facilitar el proceso de aprendizaje, el cual puede ser diferente respecto a la relevancia de los juicios sobre los documentos, es decir, incluso si utilizamos el mismo formato de juicios, se pueden obtener distintas etiquetas *ground truth* para los diferentes enfoques.

A continuación se describen de manera general los tres enfoques respecto a los componentes de ML:

Enfoque por Puntos:

- Espacio de entrada: contiene los vectores de atributos de cada par consulta-documento.
- Espacio de salida: representa el grado de relevancia único de cada documento.
- Espacio de hipótesis: se consideran funciones que toman los vectores de atributos y predicen un valor de relevancia del documento respecto a la consulta.
- Función de pérdida: se corresponde con la de problemas de regresión, clasificación o regresión ordinal, la cual evalúa la precisión de la etiqueta *ground truth* para cada documento.

Enfoque por Pares:

- Espacio de entrada: contiene pares de documentos representados por sus vectores de atributos.
- Espacio de salida: representa los órdenes relativos de los pares de documentos con respecto a la consulta en el *ranking* esperado.
- Espacio de hipótesis: contiene funciones capaces de ordenar correctamente pares de documentos respecto a su relevancia frente a una consulta.
- Función de pérdida: mide la inconsistencia relativa del orden entre los pares de documentos. Tienden a cuantificar el número de pares mal ordenados.

Enfoque por Listas:

- Espacio de entrada: contiene todo el grupo de documentos relacionados con la consulta q.

- Espacio de salida: existen dos tipos, uno que contiene los grados de relevancia de todos los documentos relacionados con una consulta, y el otro que contiene la lista ordenada de los documentos.
- Espacio de hipótesis: contiene funciones multi-variables que operan en un grupo de documentos, y predicen sus grados de relevancia.
- Función de pérdida: mide la diferencia (no correspondencia) entre la lista ordenada dada por la hipótesis (permutación de predicción) y la lista *ground truth*, usando en la mayoría de los casos métricas de evaluación de RI.

Por otro lado, también se han encaminado algunos trabajos [48, 49] muy puntuales que no pertenecen a ninguna de estas tres categorías mencionadas, pero que revisten una utilidad práctica y teórica en el campo del L2R.

2.6.1. Enfoque por Puntos (*Pointwise Approach*)

Cuando usamos las tecnologías de ML para resolver problemas de *ranking*, probablemente la manera más sencilla es comprobar si los métodos de aprendizaje existentes pueden ser directamente aplicados. Esto es exactamente lo que el enfoque por puntos hace. Al aplicarlo, asumimos que el grado de relevancia exacta de cada documento es lo que se está prediciendo. En esencia, esta primera categoría de enfoque o aproximación por puntos (*Pointwise Approach*), transforma la ordenación en una regresión, clasificación o regresión ordinal sobre documentos simples.

A continuación se enuncian los algoritmos más representativos en cada una de las subcategorías correspondientes:

1. Algoritmos basados en Regresión: en esta subcategoría, el problema de *ranking* se reduce a un problema de regresión, al considerar el grado de relevancia como números reales. Uno de los primeros ejemplos fue el trabajo propuesto por Fuhr [50] en el que se describe un enfoque para el desarrollo de funciones de recuperación polinómicas óptimas, capaz de producir estimaciones de probabilidad de relevancia. Por otro lado, tenemos que en [51] los autores plantean una aproximación de la optimización de las puntuaciones del criterio DCG (*Discounted Cumulative Gain*) a través de la minimización de errores de regresión convexa. Otra de las propuestas interesantes es

la de Kramer et al. [52] la que está encaminada a modificar al algoritmo de árbol de inducción S-CART (*Structural Classification and Regression Trees*) adaptándolo como algoritmo de aprendizaje para afrontar problemas de clasificación ordinal y así aplicarlo a la RI.

2. Algoritmos basados en Clasificación: estos algoritmos no consideran la etiqueta *ground truth* como un valor cuantitativo. Esto es mas razonable que los algoritmos basados en regresión. Como ejemplos de trabajos típicos en este sentido, tenemos el de Nallapati [53] en el que dicho autor explora la aplicabilidad de clasificadores discriminativos para la RI, con lo cual, compara el rendimiento de dos modelos discriminativos (modelo de máxima entropía y modelo de soporte vectorial) con un modelo generativo (modelado del lenguaje). También está el método McRank [54] que usa clasificación múltiple en base a árboles de decisión de menor calidad que son aprendidos de forma iterativa y combinados para mejorar las predicciones anteriores, *gradient boosting tree algorithm*.

3. Algoritmos basados en Regresiones Ordinales: estos algoritmos toman la relación ordinal entre las etiquetas *ground truth* durante el aprendizaje del modelo de *ranking*. Si hacemos una correlación de la regresión ordinal con la clasificación tenemos que es similar a la predicción multiclases, pero considerando que las clases tienen una estructura ordinal. A partir del año 1994, con el uso de técnicas de aprendizaje automático, se desarrollan nuevos y diversos métodos aproximados de regresión ordinal, que pueden ser agrupados en tres subcategorías fundamentales:

    a) Aprendizaje por múltiples umbrales: en esta subcategoría se suele dividir el espacio en regiones paralelas igualmente clasificadas, donde todas las instancias que se ubiquen en la misma zona (entre dos umbrales delimitadores) le serán asignadas la misma etiqueta de relevancia. El objetivo que persigue este tipo de aprendizaje es formular un modelo capaz de predecir una puntuación sobre una instancia entrada, minimizando la violación del umbral de predicciones. En este sentido, los mejores métodos aproximados referenciados en la literatura fueron obtenidos utilizando: procesos gausianos [55], redes neuronales (método

RankProp y el *Multitask Learning* (MTL) [56]), y modelos de soporte vectorial y perceptrones (Pranking [57], entre otros [58].

b) Aprendizaje por múltiples clasificadores: esta subcategoría tiene como objetivo contar con varios criterios para mejorar la predicción final, que resultaría en una combinación (por ejemplo, suma) de las predicciones de cada clasificador. Uno de los modelos clave dentro de esta variante lo constituye el método *Multiple Hyperplane Ranker* (MHR) [20]. Este algoritmo usa técnicas de SVM y está basado en la estrategia de divide y vencerás. Durante el aprendizaje emplea múltiples hiperplanos para rankear las instancias de entrenamiento y finalmente agrega todos estos resultados para conformar el *ranking* deseado. MHR contempla al método Ranking SVM (RSVM) [59] como un caso especial.

c) Optimización de preferencias a pares: es considerada como la más popular, dentro de las subcategorías de regresión ordinal, por su amplio uso en aplicaciones de RI. Su objetivo se basa en obtener un nuevo conjunto de entrenamiento donde las instancias sean pares de documentos y organizados de forma positiva, es decir, el primer documento es más relevante que el segundo documento. A partir de entonces se debe lograr un modelo de aprendizaje que minimice las discordancias a pares (predecir que un documento es más relevante que otro cuando no lo es). Los mejores métodos referenciados por la literatura en este sentido, fueron concebidos con el uso de redes neuronales, técnicas de *boosting*, y SVMs. Entre los más representativos están: RankNet [18], RankBoost [17], Ranking SVM (RSVM) [59] y SVM-perf [3]. También, Cohen et al. en [60] proponen una aproximación de dos etapas para aprender a ordenar instancias basándose en una función de preferencia binaria. Long y Servedio en [61] muestran como adaptar una técnica de *boosting* de clasificación para mejorar el rendimiento de una función de *ranking* usando como medida de calidad el área bajo la curva ROC (*Receiver Operating Characteristic*). Herbrich et al. [62] presentan un nuevo algoritmo de aprendizaje que se basa en un mapeo de objetos a valores de utilidad escalar, siguiendo un enfoque similar a los métodos de soporte vectorial.

A pesar de los resultados obtenidos en el enfoque por puntos, no han sido bien consideradas las propiedades de las medidas de evaluación de RI. El hecho radica en que se ignora que algunos documentos están asociados con la misma consulta y otros no. Cuando el número de documentos asociados varía grandemente para diferentes consultas, la función de pérdida global será dominada por aquellas consultas con un gran número de documentos. Otro de los problemas de esta aproximación es que la posición de los documentos en la lista ordenada es invisible a la función de pérdida. La función de pérdida en el enfoque por puntos puede inconsistentemente dar demasiado énfasis aquellos documentos sin importancia (los cuales son ordenados en bajas posiciones en el resultado o *ranking* final).

2.6.2. Enfoque por Pares (*Pairwise Approach*)

La segunda categoría se denomina enfoque o aproximación por pares (*Pairwise Approach*), en la cual de manera general se transforma o formaliza el problema de la ordenación en una clasificación binaria sobre pares de documentos. Este enfoque, no se centra en predecir con exactitud el grado de relevancia de cada documento, sino que se preocupa por el orden relativo entre dos documentos. En este sentido, está más cerca del concepto de *ranking* que el enfoque *Pointwise*. En este enfoque de dos a dos, el problema de *ranking* se reduce a un problema de clasificación sobre pares de documentos. Es decir, el objetivo del aprendizaje es minimizar el número de pares de documentos mal clasificados, por lo tanto, el fin es hacer predicciones positivas en esos pares, en el cual el primer documento es más relevante que el segundo documento, y hacer predicciones negativas sobre otros pares. En el caso extremo y deseado, si todos los pares de documentos están clasificados correctamente, todos los documentos serán correctamente ordenados. Hay que tener en cuenta que los pares de documentos no son independientes, lo cual viola el principio básico de clasificación. En este caso, aunque todavía se pueden utilizar algoritmos de clasificación para aprender el modelo del *ranking*, es necesario un esquema diferente para analizar la generalización del proceso de aprendizaje.

Los algoritmos más representativos de esta categoría son listados a continuación:

- RankNet [18] es un método de L2R que es capaz de entrenar una función de *ranking* sobre pares de documentos, usando una formulación de red neuronal sustentada en una función simple de costo probabilístico junto con gradiente descendente.

- RankBoost [17] es un método de L2R que, basado en un enfoque *boosting*, intenta resolver el problema de preferencias a pares de forma directa, en lugar de resolver un problema de regresión ordinal.

- Ranking SVM (RSVM) [59] es un método típico que desarrolla la tarea del *ranking* utilizando técnicas de SVM para clasificar pares de instancias. Este modelo es construido para identificar las relaciones de orden que se establecen entre los dos documentos de cualquiera de las instancias en un conjunto de entrenamiento.

- IRSVM [8] es un método que tomando como base a RSVM, modifica su función de pérdida conduciendo una optimización basada en gradiente descendiente y programación cuadrática.

- FRank [19] es un método que utiliza un modelo aditivo generalizado similar a *boosting* para minimizar una función de pérdida que se propone también en el mismo trabajo y que le denominan *Fidelity Loss*. Esta medida es una adaptación de la distancia métrica *Fidelity* utilizada en física para medir la diferencia entre dos estados de un *quantum*.

- El método MHR [20] explicado en la sección anterior también puede ser incluido en esta categoría.

- Entre otros métodos de interés como el LDM [63], GBRank [64], QBRank [65] y MPRank [66].

Alguno de estos métodos también están enmarcados dentro de la tercera subcategoría (Optimización de preferencias a pares) perteneciente a la regresión ordinal.
En general, aunque de esta categoría han emanados métodos robustos teóricamente, existen problemas que se manifiestan desde su propia concepción. Por ejemplo, cuando el juicio

de relevancia está dado en términos de múltiples categorías ordenadas, convertir esto en preferencia a pares da lugar a falta de información acerca de la más fina granularidad en el juicio de relevancia. La distribución del número de pares de documentos es más sesgada que la distribución del número de documentos, con respecto a las diferentes consultas. Este enfoque es más sensible a etiquetas ruidosas que el enfoque puntual y además, la mayoría de los algoritmos de *ranking* basados en esta aproximación no han considerado en su concepción la posición de los documentos en los resultados finales del *ranking*, lo cual contribuiría a mejoras en la disposición y actuación de dichos modelos de *ranking*.

2.6.3. Enfoque por Listas (*Listwise Approach*)

La tercera categoría hace referencia a la aproximación por listas (*Listwise Approach*), en la cual se minimiza directamente una función de pérdida definida sobre una lista de documentos. Este enfoque por listas puede ser dividido en dos subcategorías.
En la primera, el espacio de salida contiene la permutación de los documentos asociados con la consulta y la función de pérdida mide la diferencia entre la permutación dada por la hipótesis y la permutación del *ground truth*. Para la segunda subcategoría, el espacio de salida contiene los grados de relevancia de todos los documentos relacionados con una consulta, y la función de pérdida es definida directamente o en la aproximación de una medida de evaluación de RI.
A continuación se presentan tales vertientes y sus métodos asociados:

- Minimización de las Pérdidas en las Listas de *Ranking*: en esta subcategoría se minimiza una función de pérdida definida sobre las permutaciones, las cuales son diseñadas considerando las propiedades de la ordenación en la RI. Específicamente, la función de pérdida mide la inconsistencia entre la salida del modelo de *ranking* y la permutación *ground truth* de todos los documentos. Las propuestas algorítmicas más relevantes dentro de esta vertiente son: ListNet [67], ListMLE [24], RankCosine [68] y BoltzRank [69].

- Optimización Directa de las Medidas de Evaluación en RI: en esta segunda línea investigativa, el modelo de *ranking* aprende directamente de la optimización de una

de las medidas de evaluación de RI (mayormente MAP o NDCG), o por lo menos, de alguna función aproximada a tales medidas. En este caso, la medida de RI evaluará la precisión alcanzada según la disposición en el *ranking* del *ground truth* de los documentos devueltos por el modelo. Como algoritmos más representativos podemos mencionar los siguientes: AdaRank [9], SVMmap [21], SVMperf [3], SVMstruct [70], SoftRank [22], LambdaRank [23] [71], CCA [72] y RankGP [4]. En la Sección 2.6 se aborda detalladamente esta subcategoría.

2.6.4. Evolución histórica

En la Figura 2-4, se muestra una línea de tiempo del L2R, donde se refleja el momento histórico y la tendencia categórica de la mayoría de los métodos mencionados anteriormente. Se puede apreciar que en estos últimos años ha habido una tendencia justificable hacia desarrollar métodos de L2R sustentados en los principios del enfoque por listas. En este enfoque además, de manera acertada, se toman todos los documentos relacionados con la misma consulta como instancia de aprendizaje y la posición en el *ranking* es siempre visible para la función de pérdida.

A pesar de esto, y de que los modelos de esta categoría superan diversos problemas de los enfoques anteriores, nos encontramos con diversos problemas a resolver. Primeramente, la optimización directa de medidas de rendimiento no es una tarea trivial. Las medidas de evaluación, tales como NDCG son discontinuas y no diferenciables debido a que dependen de las posiciones del *ranking* y es un reto optimizar tales funciones objetivo, ya que la mayoría de las técnicas de optimización referenciadas en la literatura se han desarrollado para manejar casos continuos y diferenciables. También es notorio mencionar el problema de la complejidad (algorítmica) y que el uso de la información sobre la posición en el *ranking* es aún insuficiente.

En general, la mayor diferencia entre los tres enfoques principales de L2R, está dada por la función de pérdida. Las funciones de pérdidas, son principalmente usadas para guiar el proceso de aprendizaje, mientras que la evaluación del modelo de *ranking* aprendido se basa en las medidas de evaluación de RI.

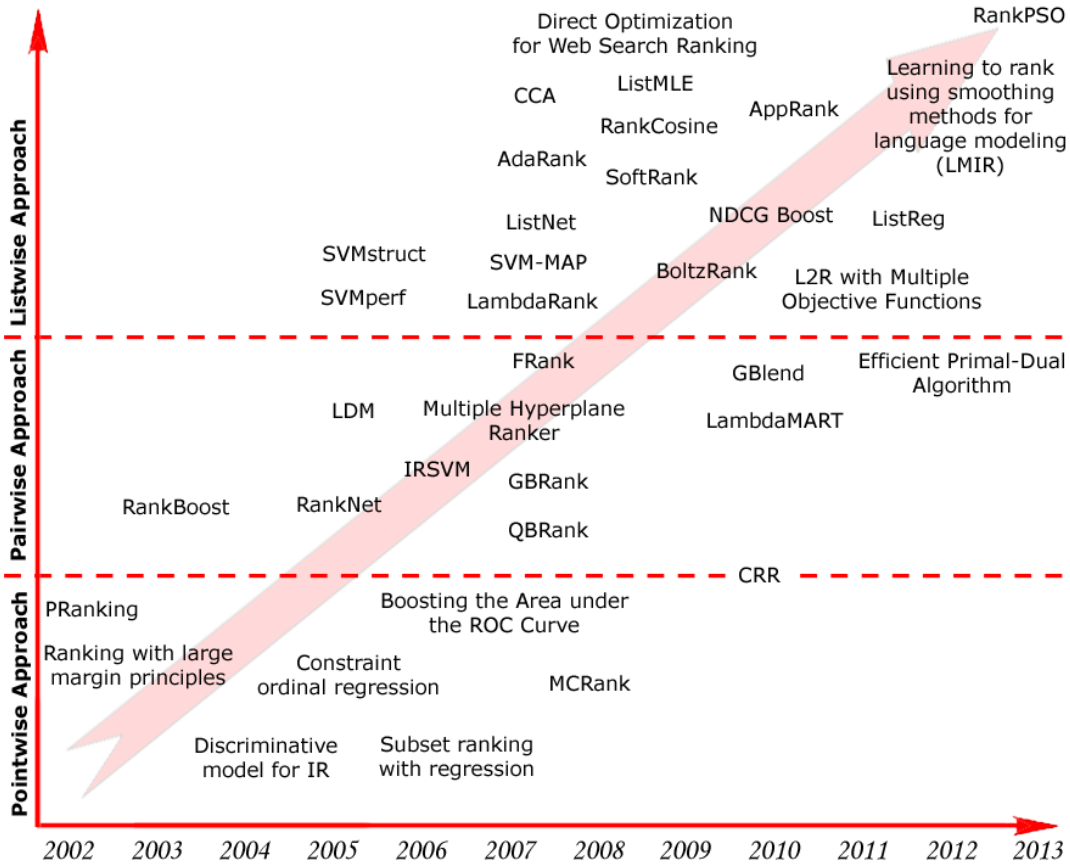


Figura 2-4: Línea de tiempo y tendencia categórica del aprendizaje de la ordenación.

Por tal razón, la optimización directa de medidas de rendimiento resulta en una línea prometedora, que bien guiada nos podrá llevar a mejores rendimientos en la tarea del L2R. En el siguiente apartado se hará alusión a las principales vertientes y métodos pertenecientes a esta categoría.

2.7. Optimización directa de medidas de rendimiento

A partir del año 2005, muchos investigadores del área han encaminado sus esfuerzos a desarrollar métodos aproximados que optimicen directamente medidas de evaluación empleadas en RI. Lo cual ha resultado en una tarea compleja debido a que tales medidas se basan en la posición del *ranking*, presentan discontinuidades y son no diferenciables. Los modelos en los que se ha venido trabajando para enfrentar este desafío pueden ser agrupados en tres nuevas categorías dentro del L2R:

1. Minimizar una función de pérdida que acote superiormente a una función de pérdida básica definida sobre las medidas de RI.

2. Aproximar las medidas de evaluación de RI mediante funciones fáciles de manipular.

3. Usar tecnologías especialmente diseñadas para optimizar medidas de rendimiento de RI no suaves (*non-smooth*).

2.7.1. Minimización de funciones de pérdida que acoten superiormente a una función de pérdida básica definida sobre las medidas de RI

Dentro de esta categoría se ha trabajado en varias vertientes haciendo uso de técnicas de *boosting*, *structural SVMs*, entre otras propuestas basadas en métodos estadísticos y de ML. Debido a dificultades computacionales, pocas técnicas de aprendizaje se han desarrollado para optimizar directamente una medida de evaluación, como por ejemplo, MAP. Uno de los trabajos más reconocidos en esta subcategoría es el SVM-map de Yue et al. [21], donde se presenta una aproximación general para aprender funciones de ordenación que maximicen el rendimiento de MAP. Específicamente, se presenta un algoritmo utilizando SVMs que optimiza globalmente una versión más laxa de la función de pérdida (*hinge loss relaxation*). Esta aproximación simplifica el proceso de obtención de funciones de ordenación con un alto rendimiento de MAP, evitando pasos intermedios y heurísticas adicionales.

Por otro lado, tenemos el trabajo de Metzler y Croft [73] donde se describe un modelo de recuperación que basado en el campo aleatorio de Markov implementa tres variantes para optimizar la medida MAP, capturando la independencia total, la dependencia secuencial, y la dependencia total entre los términos de las consultas. Este modelo muestra como rasgos arbitrarios del texto pueden ser usados para generalizar la idea de la co-ocurrencia, y demuestra que los modelos dependientes son más eficaces para las colecciones más grandes que en las más pequeñas, y que la incorporación de varios tipos de rasgos dentro de un modelo dependiente mejorará su eficacia. Sin embargo, este algoritmo aproximado

no encuentra una solución óptima y resulta caro computacionalmente. Por su parte, la aproximación de Yue et al. mencionada, a diferencia de la propuesta de Metzler y Croft [73], puede ocuparse de miles de rasgos, también aprende un modelo lineal, es mucho más eficaz en la práctica y además, encuentra eficientemente una solución óptima global para una función que acota directamente la medida MAP.

Otro de los métodos, que brinda un aporte novedoso es el propuesto por Chapelle et al., en [74], donde se enfoca el problema de la ordenación como un problema de aprendizaje de salida estructurada. Este algoritmo trata directamente de perfeccionar una medida de calidad compleja como el NDCG.

Con otro enfoque, Zheng et al. en [65] presentaron un método general basado en *boosting* que amplia el gradiente funcional para optimizar una función de pérdida compleja que ha sido encontrada en varios problemas de aprendizaje automático. Esta aproximación está basada en la optimización de cotas superiores cuadráticas de funciones de pérdidas que permiten representar un análisis riguroso de convergencia del algoritmo.

En el 2008, Xu et al. en [5], realizan todo un estudio teórico sobre los métodos que minimizan una función de pérdida que acote superiormente una función de pérdida básica definida sobre las medidas de RI. Proponen además, un nuevo algoritmo de optimización directa denominado PermuRank que minimiza eficientemente una de las cotas de tipo dos como función de pérdida de una manera voraz (*greedy*).

Otra de las propuestas interesantes es la de Xu et al. en [9], que inspirados en el trabajo de AdaBoost (*Adaptive Boosting*) [75] para clasificación, proponen un novedoso algoritmo de L2R para la RI, referido como AdaRank, el cual, minimiza una función de pérdida definida directamente sobre cualquiera de las medidas de evaluación de RI. Este algoritmo se basa en una técnica de *boosting* para el aprendizaje de un modelo de *ranking*. Inicialmente AdaRank define una distribución de pesos sobre las consultas del conjunto de entrenamiento. Estos pesos indican cuán complejo resulta ordenar correctamente los documentos de cada consulta. En un principio todas las consultas tendrán un mismo peso básico, pero en cada nueva iteración el algoritmo incrementará los pesos de aquellas consultas que no fueron ordenadas muy bien por el modelo creado hasta el momento. Como resultado en las próximas rondas de aprendizaje la creación de "*weak rankers*" estará más enfocada en

estas consultas más "duras". En la implementación como "*weak ranker*" se elige el rasgo que obtuvo el mayor rendimiento ponderado entre todos los rasgos. La creación de "*weak rankers*" continúa de esta manera, y el proceso de aprendizaje consiste en ir seleccionando repetidamente rasgos e ir combinándolos linealmente. Finalmente los rasgos que al concluir la fase de entrenamiento no han sido seleccionados tendrán un peso igual a cero. Este algoritmo para definir su función a optimizar considerando una medida de evaluación de RI, parte de la idea de maximizar la siguiente expresión sobre el conjunto de entrenamiento:

$$\max_{f \in F} \sum_{i=1}^{m} E(\pi(q_i, d_i, f), y_i), \tag{2-3}$$

donde F es el conjunto posibles de funciones de *ranking*, m la cantidad de consultas. El significado de los demás términos fue detallado en la Sección 2.4. Esta expresión es equivalente a minimizar la pérdida de:

$$\min_{f \in F} \sum_{i=1}^{m} (1 - E(\pi(q_i, d_i, f), y_i)), \tag{2-4}$$

y, E, es una función no continua y difícil de manipular. AdaRank incorpora una función exponencial como cota superior de esta perdida:

$$\min_{f \in F} \sum_{i=1}^{m} \exp\{-E(\pi(q_i, d_i, f), y_i)\}, \tag{2-5}$$

considerando matemáticamente que siempre se cumple que $e^{-x} \geqslant 1 - x$ para todo $x \in \mathbb{R}$. El proceso de aprendizaje de AdaRank ha demostrado ser más eficiente que el de la mayoría de los algoritmos de aprendizaje.

2.7.2. Aproximación de medidas de rendimiento de RI mediante funciones fáciles de manipular

La segunda categoría que aborda este apartado está referida a aproximar las medidas de evaluación de RI mediante funciones fáciles de manipular. En esta vertiente se han desarrollado pocos trabajos, considerando que tales medidas son difíciles de aproximar por los comportamientos no suaves, discontinuos y complejos que presentan.

A pesar de esto, Olivier Chapelle introduce en su artículo: "*Direct Optimization for Web Search Ranking*" [76], dos nuevos algoritmos de aprendizaje de la ordenación, desarrollados con el objetivo de optimizar directamente la medida de evaluación NDCG. Uno de los métodos usa una aproximación de NDCG, y el otro está basado en un aprendizaje de rendimiento estructurado.

En esta misma línea, los autores Taylor, Guiver, Robertson y Minka publican un método denominado Softrank [22], en el cual proponen una aproximación suavizada de NDCG, con la que obtienen resultados alentadores.

Por su parte, David y Tong en [13] realizan un estudio de cómo el criterio DCG enfatiza la calidad de los elementos rankeados en altas posiciones del *ranking*, y a partir de esta premisa, exponen un método en el cual deducen cotas (límites) que relacionan la optimización de las puntuaciones de DCG con la minimización de errores de regresión convexa.

2.7.3. Tecnologías especialmente diseñadas para optimizar medidas de rendimiento de RI no suaves

La tercera categoría dirige su atención a poder usar tecnologías especialmente diseñadas para optimizar medidas de evaluación de RI no suaves. En esta línea, los investigadores han trabajado en tres subcategorías fundamentales: (1) Aproximaciones Suaves, (2) Aproximaciones Suaves utilizando Programación Genética y (3) Aproximaciones Suaves con Gradiente Descendiente.

(1) Aproximaciones Suaves.

Los autores Metzler, Croft y McCallum, en [77], plantean una técnica de optimización simple que no hace supuestos sobre la función objetivo subyacente. En este artículo, se investigan las propiedades lineales y monótonamente lineales de las funciones de puntuación cuando se usan junto a la clase de métricas de evaluación basadas en el ordenamiento. Donde se muestra como los parámetros de las funciones lineales de puntuaciones se transforman a un espacio multinomial n-dimensional (*multinomial manifold*). Esto no sólo proporciona teóricamente una representación elegante del espacio de parámetros, sino que también reduce significativamente el número de extremos locales garantizando acotar el extremo global.

Otro método importante a considerar es el propuesto por Burges et al. en [71]. En este trabajo, los autores presentan un algoritmo simple y flexible denominado LambdaRank, el cual trabaja con funciones de costo implícitas evitando que las derivadas del costo con respecto al modelo de parámetros sean cero o estén indefinidas. Hay que precisar además, que aunque este método va dirigido hacia la ordenación, puede ser extendido a cualquier función de costo no suave y multivariable.

(2) Aproximaciones Suaves utilizando Programación Genética.

Dentro de esta vertiente es conveniente analizar un artículo publicado en el año 2007 por Almeida, Gonalves, Cristo y Calado [78]. En este trabajo se propone un método basado en programación genética para descubrir funciones de ordenación adaptadas a colecciones. Esta Aproximación de Componente Combinado (*Combined Component Approach*, CCA) está basada en la combinación de varios componentes de términos ponderados (por ejemplo, frecuencia del término, frecuencia de la colección, normalización, entre otros) extraídos a partir de funciones de ordenación históricamente conocidas. A diferencia de trabajos relacionados, los términos de CCA no están basados en información estadística simple de una colección documental, sino en componentes significativos, eficaces y probados.

Fan, Pathak y Wallace en su artículo [79] exponen un esquema representativo de una función de ordenación no lineal y comparan este nuevo diseño con el modelo del espacio vectorial mostrando teóricamente que el nuevo esquema representativo incluye el modelo de espacio vectorial tradicional como un caso especial y además, permite una flexibilidad adicional en la ponderación de los términos.

Por otro lado, es vital mencionar el método RankGP de los autores Yeh, Lin, Ke y Yang [4]. RankGP emplea programación genética para aprender una función de ordenación combinando varios tipos de evidencias en la RI, incluyendo rasgos de contenido, de estructura y rasgos independientes de la consulta.

(3) Aproximaciones Suaves con Gradiente Descendiente.

Dentro de esta última subcategoría, se impone el trabajo de Snelson y Guiver [80], publicado en el año 2008, donde se dirige el asunto del aprendizaje de la ordenación en la recuperación de documentos utilizando modelos *Thurstonian* basados en algunos procesos gausianos. Los modelos *Thurstonian* representan cada documento para una consulta dada

como una distribución de probabilidad en un espacio de puntos; estas distribuciones sobre puntos dan lugar naturalmente a distribuciones sobre el ordenamiento de los documentos. Sin embargo, en general no se observan ordenamientos con los cuales entrenar el modelo; más bien, cada documento en el conjunto de entrenamiento se juzga para tener un nivel de relevancia particular, por ejemplo: "malo", "aceptable", "bueno", o "excelente". El rendimiento del modelo es entonces evaluado usando medidas de RI tales como NDCG.

2.8. Colecciones estándar de prueba en L2R

Las colecciones de datos de LETOR[6] [43], publicadas por *Microsoft Research Asia*, comprenden los primeros *datasets* de referencia para la investigación en el ámbito del aprendizaje de la ordenación. Este paquete de *datasets* contiene características estándar, juicios de relevancia, partición de datos, herramientas de evaluación, y resultados obtenidos por métodos de referencia en este campo. A partir del año 2007 y hasta la actualidad, LETOR ha lanzado las cuatro versiones siguientes:

1. La Versión 1.0, liberada en abril de 2007.

2. La Versión 2.0, que constituye una actualización basada en la Versión 1.0, se lanza en diciembre de 2007.

3. La Versión 3.0 se publica en diciembre de 2008 y constituye una de las versiones más utilizadas. Esta contempla las colecciones OHSUMED y ".gov".

4. La Versión 4.0 liberada en julio de 2009, utiliza la colección de páginas web ".gov2" y dos conjuntos de consultas a partir de millones de consultas provenientes de TREC 2007 y TREC 2008. Los datasets propuestos son MQ2007 con un total de 1700 consultas y MQ2008 con alrededor de 800 consultas.

OHSUMED[7] constituye la primera colección de LETOR 3.0 y consiste en un subconjunto de MEDLINE[8], base de datos de publicaciones médicas, que reúne artículos de 270 revistas médicas correspondientes al período de 1987-1991. OHSUMED presenta 348.566

[6]Disponibles en: http://research.microsoft.com/en-us/um/beijing/projects/letor/
[7]Disponible en: http://research.microsoft.com/en-us/um/beijing/projects/letor/LETOR3.0/OHSUMED.zip
[8]Disponible en: http://www.ncbi.nlm.nih.gov/pubmed

referencias o registros y 106 consultas. Para cada consulta, existe un número de documentos asociados. Los grados de relevancia de los documentos con respecto a las consultas son dados por humanos, en tres niveles: definitivamente relevante "2", parcialmente relevante "1", y no relevante "0". En OHSUMED un total de 16.140 pares de consulta-documento están etiquetados con juicios de relevancia y 45 rasgos son extraídos.

Por otro lado, la colección ".gov" reúne seis *datasets*: TD2003, TD2004, HP2003, HP2004, NP2003 y NP2004. Existe un total de 1.053.110 documentos html en esta colección, junto con 11.164.829 hipervínculos. El corpus documental de TD2003 y TD2004 está dado a partir del *topic distillation task* de TREC 2003 and TREC 2004. Para el caso de los *datasets*: HP2003, HP2004, NP2003 y NP2004, cada corpus documental fue concebido respectivamente según *homepage finding* 2003, *homepage finding* 2004, *named page finding* 2003 y *named page finding* 2004.

Específicamente, TD2003 agrupa 50 consultas; cada uno de los *datasets* TD2004, HP2004 y NP2004 tienen 75 consultas y HP2003 y NP2003 poseen 150 consultas. Para cada consulta, se tienen aproximadamente 1.000 documentos asociados. Cada par consulta-documento está etiquetado con un juicio binario: relevante o irrelevante. Estos *datasets* tienen 64 rasgos, que incluyen rasgos de bajo nivel como la frecuencia de los términos (tf), frecuencia inversa del documento (idf) y longitud del documento (dl), así como rasgos de alto nivel como BM25, LMIR, PageRank y HITS.

Más recientemente, en junio de 2010, *Microsoft Research*[9] liberó dos colecciones de referencia de gran tamaño para la investigación en el área del L2R en RI. Estas dos últimas colecciones son publicadas como: MSLR-WEB30k que consta de más de 30.000 consultas y MSLR-WEB10K con aproximadamente 10.000 consultas, estas últimas constituyen un muestreo aleatorio de la primera colección. Los juicios de relevancia de estos *datasets* obtenidos a partir del motor de búsqueda web comercial Microsoft Bing[10], tienen 5 valores de 0 (irrelevante) a 4 (perfectamente relevante). Cada par consulta-documento está representado por un vector de rasgos de 136 dimensiones.

En la presente investigación, la mayoría de estos conjuntos de datos fueron utilizados

[9]Disponible en http://research.microsoft.com/en-us/projects/mslr/default.aspx
[10]Disponible en https://www.bing.com/

indistintamente durante las pruebas experimentales.

2.9. Conclusiones

En el presente capítulo se presentó una revisión sintetizada del estado del arte sobre el L2R para la RI, donde se han presentado y descrito una serie de métodos relevantes de L2R siguiendo un orden histórico y definiendo su desglose categórico.

Se puede constatar además, como los métodos tradiciones de *ranking* para la RI, han ido cediendo su espacio a nuevas técnicas de aprendizaje automático, todo ello propiciado por la necesidad imperante de afrontar nuevos retos y problemas de ordenación que exigen precisión, simplicidad, bajo costo computacional, disminución de la complejidad algorítmica, y finalmente el lograr la satisfacción del usuario.

Es importante resaltar, después de haber analizado las tendencias categóricas y los diferentes modelos, que los métodos de ordenación que llevan a cabo una optimización directa de medidas de evaluación, ofrecen elementos teóricos y resultados prometedores que ostentan mejoras significativas en cuanto al rendimiento alcanzado en la ordenación con respecto a las demás propuestas y estrategias.

Finalmente, fueron enunciadas también las medidas de evaluación utilizadas en este campo y aquellas colecciones estándar de datos que han sido establecidas como referencia internacional.

PARTE II

Estudio Bibliométrico sobre Learning to Rank

Capítulo 3

Estudio bibliométrico de la producción científica sobre Aprendizaje de la Ordenación

"La ciencia es un conocimiento organizado".
— Herbert Spencer

3.1. Introducción

El presente capítulo aborda detalladamente un estudio métrico desarrollado para evaluar la producción científica sobre el Aprendizaje de la Ordenación en el período 2000-2013. Esta investigación tiene como objetivo principal ofrecer a la comunidad científica interesada en el ámbito del L2R, información fidedigna y actualizada en cuanto al comportamiento y las tendencias de este campo multidisciplinar. Con esta premisa, se realizó un análisis del marco teórico conceptual sobre los estudios métricos de la información relacionados con las ciencias de la información y se llevó a cabo un proceso de búsqueda y recuperación de registros bibliográficos relevantes y pertinentes al L2R en la Base de Datos Scopus[1], la cual constituye una de las bases de datos de mayor prestigio e impacto internacional. Los registros obtenidos fueron procesados en correspondencia con una serie de indicadores métricos seleccionados para el estudio, tanto de carácter unidimensional como multidimensional. Finalmente, se analizó cada uno de estos indicadores con el apoyo de la visualización de gráficos y mapas topológicos que describen cuantitativa y cualitativamente el comportamiento de la actividad científica y tecnológica del L2R, así como la estructura y dinámica social de los que producen y utilizan su literatura.

[1]Disponible en http://www.scopus.com/home.url

3.2. Los Estudios Métricos en las Ciencias de la Información

El desarrollo de la ciencia contemporánea y la informática, sin lugar a dudas, se debe en parte, a la penetración de métodos y modelos matemáticos en todas las esferas del conocimiento. "Este proceso que cada vez toma mayor fuerza en las ciencias en general y las sociales en particular, se conoce como la "Matematización del conocimiento científico" [81].

A pesar de la existencia de nuevas disciplinas instrumentales, surgidas con el propio desarrollo científico-tecnológico de los últimos años, la mayoría de las investigaciones toman como punto de referencia la Bibliometría, la Cienciometría y la Informetría como disciplinas instrumentales de la Bibliotecología, la Cienciología y la Ciencia de la Información, respectivamente.

La aparición del *Science Citation Index* (SCI) y los productos como el *Journal Citation Report* (JCR), los mapas de la ciencia y el *Web of Science* o *Web of Knowledge*, han influido significativamente en el desarrollo de la ciencia moderna.

Entre las principales disciplinas métricas se encuentra la Bibliometría, cuya definición moderna (*bibliometrics*) se le atribuye a Alan Pritchard en un trabajo publicado en 1969. Esta definición establece que la Bibliometría es "la aplicación de los métodos estadísticos y matemáticos dispuestos para definir los procesos de comunicación escrita y la naturaleza y el desarrollo de las disciplinas científicas mediante el recuento y análisis de dicha comunicación" [82]. En el último siglo se han usado una gran variedad de medidas bibliométricas [83].

Otra de las disciplinas métricas es la Cienciometría, cuyo término en un principio, se refería sólo a la aplicación de métodos cuantitativos a la historia de la ciencia y el progreso tecnológico. Dicha disciplina utiliza métodos matemáticos para el estudio de la ciencia y a la actividad científica en general, además de medir el nivel de desarrollo y el aporte de la ciencia a las diferentes esferas de la sociedad. Según [81] la Cienciometría "es una disciplina métrica perteneciente a la Cienciología, que utiliza en la metría del conocimiento, variables e indicadores métricos de la información documentada, por constituir las publicaciones científicas el canal más utilizado en la transferencia del conocimiento científico".

Como consecuencia del desarrollo de la ciencia y la técnica, se amplió el alcance de las disciplinas métricas cobrando importancia estratégica para la búsqueda de oportunidades tecnológicas, así como para la evaluación de programas de I+D.

Los métodos bibliométricos y cienciométricos son muy similares, a veces perfectamente idénticos. El primero analiza el tamaño, crecimiento y distribución de la bibliografía científica (libros, revistas, etc.), con el objetivo de mejorar las actividades de información, documentación y comunicación científica. El segundo, por su parte, analiza los aspectos de generación y programación de la literatura científica para llegar al mejor entendimiento de los mecanismos de la investigación científica, considerada como una actividad social, así como la estructura y dinámica social de los que producen y utilizan esta literatura.

Con la aplicación de la Matemática y la Estadística al objeto de estudio de la Ciencia de la Información, surgió la Informetría como su disciplina instrumental. El autor Morales [84] plantea que esta disciplina métrica se basa en el estudio de métodos matemáticos - estadísticos y sus correspondientes modelos, así como su aplicación al análisis cuantitativo de la estructura, las propiedades de la información científica y los patrones y las leyes de los procesos de comunicación social, incluida la identificación adecuada de sus leyes.

Morales [85] señala además que: "La Informetría no sólo permite revelar tendencias, regularidades y leyes informacionales, sino que también permite optimizar la toma de decisiones".

En sentido general, podemos considerar la Informetría como disciplina integradora muy importante a la hora de seleccionar y adquirir información, pues permite conocer las publicaciones más importantes y/o recién surgidas dentro de los diferentes campos de investigación, así como autores o instituciones que más trabajan una temática, lo que permite realizar una mejor cobertura informativa sin gasto adicional en material que no satisfaga a los usuarios. El campo de aplicación no se circunscribe a un área determinada sino que son disímiles sus aplicaciones prácticas y considera aspectos cuantitativos de la comunicación informal o hablada al igual que de la registrada. [86]

Con respecto a la convergencia de estas disciplinas, algunos autores han optado por hablar de BIC (Bibliometría, Informetría y Cienciometría). Gráficamente estas relaciones se representan de la manera siguiente:

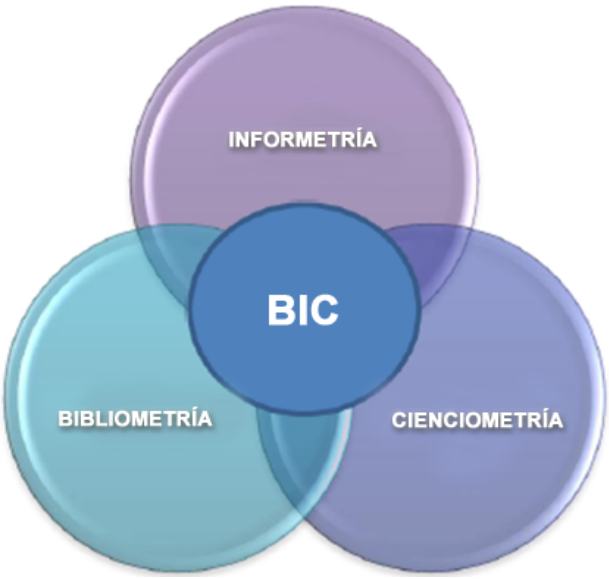


Figura 3-1: Convergencia de las disciplinas métricas.

Como se representa en la Figura 3-1 la BIC aparecería en el punto de confluencia entre distintas disciplinas caracterizada por sus métodos y los fenómenos que ha desvelado y cuyas variaciones en los objetivos planteados la acercarían respectivamente a la Biblioteconomía, a la Documentación o a la Ciencia de la Ciencia. [87]

Según Araújo [88]: "Los términos anteriormente analizados se relacionan entre sí por representar una ciencia general, la ciencia métrica, esto contribuye a que en múltiples ocasiones los modelos, indicadores, índices y demás mediciones se utilicen indistintamente en una u otra ciencia, pero se distinguen por su objeto de estudio y los objetivos que persiguen sus resultados".

En este capítulo se han utilizado básicamente métodos de la Bibliometría y elementos de la Cienciometría con el objetivo de determinar cuantitativamente el comportamiento de la actividad científica, la tendencia tecnológica del L2R, así como la estructura y dinámica social de los que producen y utilizan esta literatura.

3.3. Scopus: Base de Datos bibliográfica en línea

La veracidad de los resultados que se obtienen de un estudio bibliométrico depende, en

gran medida, de la fuente de información de la cual se extraen los datos. Los trabajos publicados en las fuentes primarias suelen recopilarse en forma abreviada en las bases de datos. Es por ello que la consulta a las bases de datos apropiadas es un método adecuado para obtener información sobre las publicaciones de primer orden y constituye la base para estudiar el comportamiento y las tendencias de la producción científica en una ciencia o disciplina.

Las bases de datos de mayor prestigio e impacto internacional son *Web of Science* y *SciVerse Scopus*[2] (comúnmente llamada Scopus). Si se toma como punto de referencia el universo de revistas científicas arbitradas que componen el directorio internacional de publicaciones seriadas Ulrich's, *Web of Science* procesa sólo el 25 % de ellas, mientras que Scopus abarca el 50 %. Además, Scopus procesa el 95 % de las fuentes que ingresan al *Web of Science* [89, 90].

Scopus, creada en 2004 por Elsevier B. V., es la mayor base de datos de citas y resúmenes de literatura arbitrada y de fuentes de alta calidad en el Web. Cubre cerca de 18.000 publicaciones seriadas de más de 5.000 casas editoras; 16.500 son revistas arbitradas. Contiene más de 40 millones de registros procedentes de publicaciones seriadas (revistas y series monográficas) y comerciales. Presenta además, una extensa cobertura de materiales de conferencias (más de 3.6 millones), páginas Web en Internet (unos 318 millones) y patentes (23 millones). [91]

La retrospectividad del procesamiento de los artículos y sus referencias (necesarias para los análisis de citación) se remonta al año 1996, aunque existe una gran cantidad de artículos fuentes (es decir, sin sus referencias) de fechas anteriores.

Scopus tiene como fortaleza el área de los estudios de citación en el contexto científico internacional, por lo que ha producido gran interés entre investigadores y académicos, tanto por su cobertura documental, como por su amigable interfaz y sus múltiples funcionalidades. Además, posee herramientas para seguir, analizar y representar el comportamiento de la actividad científica, mediante el empleo de los datos de citación de los artículos y los autores. Por tal razón, se ha convertido en un fuerte competidor de los productos y servicios creados por el *Institute for Scientific Information* (ISI).

[2]Disponible en http://www.elsevier.com/online-tools/scopus

Scopus tiene como objetivo futuro convertirse en la herramienta de navegación de mayor envergadura, abarcando la literatura internacional sobre ciencias, medicina, tecnología y ciencias sociales. Un equipo editorial revisa continuamente publicaciones adicionales para su inclusión en Scopus. Se esfuerza por incluir todas las publicaciones académicas que cumplen con las normas básicas de calidad científica. Además, está ampliando la cobertura a *Book Series* (colecciones monográficas), obras de referencia y a la web, para abarcar todas las fuentes importantes de información científica.

3.4. Procedimiento general realizado para el análisis métrico

El estudio métrico se realizó utilizando como muestra los registros bibliográficos recuperados de la base de datos Scopus.

La estrategia de búsqueda se formuló de manera que permitiese recuperar todos los documentos en los que apareciera la frase en inglés "*Learning to rank*" en los campos: título, resumen y palabras clave, lo que garantizaría la relevancia de los registros recuperados. Se limitó la búsqueda a registros que fueran del período 2000-2013, lo cual resulta totalmente representativo y abarcador para determinar el comportamiento y las tendencias del Aprendizaje de la Ordenación; y constituye la muestra de la presente investigación.

Como resultado de la búsqueda se recuperaron 623 registros bibliográficos. Estos fueron exportados al programa EndNote X4[3], para conformar la base de datos bibliográfica. Se realizó un filtrado sobre dicha base de datos bibliográficos para eliminar registros duplicados. Se verificó además, que los registros bibliográficos contaran con la información requerida en los campos para poder realizar el análisis de los indicadores seleccionados. Finalmente, se generaron las listas de los campos que forman el registro bibliográfico como tal: autores, descriptores, años, tipo de referencias, etc. con la frecuencia de aparición de cada uno. Los datos de los campos de estas listas fueron ordenados alfabéticamente, y además, normalizados y rectificados en aquellos casos donde existieron homógrafos y sinónimos. Luego se contabilizaron y ordenaron la cantidad total de registros bibliográficos obtenidos y la cantidad de documentos por años de publicación, lo que ofreció una

[3]Disponible en http://www.endnote.com/pr-enx4win.asp

visión más clara de los años en los que hubo mayor actividad investigativa. De manera similar se procedió con todos los campos utilizados para el análisis de los indicadores unidimensionales. Para una mejor interpretación de estos indicadores, fueron visualizados además a través de gráficos tradicionales (líneas, barras, etc.).

Por su parte, el análisis de los indicadores multidimensionales, requirió de un procesamiento más complejo de todos los registros bibliográficos y de las funcionalidades del programa Toolinf[4]. Se confeccionaron listas de campos con su frecuencia de aparición y los códigos correspondientes, que permitieron interrelacionar las variables, a fin de realizar conteos automáticos, para generar matrices, visualizar la información procesada mediante mapas auto-organizativos (*Self-Organizing Maps*, SOM) [92] y redes sociales. Para la visualización de los indicadores multidimensionales se utilizaron los programas Viscovery SOMine[5] (mapas topológicos), Ucinet[6] y NetDraw[7] (redes sociales), que forma parte del paquete de Ucinet.

3.5. Indicadores Unidimensionales

Los indicadores de actividad o unidimensionales constituyen el primer grupo de indicadores utilizados para analizar las fuentes de información y explorar los campos científicos o del conocimiento. Los principales indicadores de este grupo son: evolución de la ciencia o disciplina científica, temáticas más tratadas, países con mayor actividad científica, productividad de los autores, impacto de la productividad de los autores y fuentes de información más relevantes. Tales indicadores se ocupan del recuento de elementos bibliográficos para determinar el tamaño y describir la producción científica y tecnológica.

3.5.1. Evolución de la temática "Learning to rank"

La evolución de la producción científica sobre L2R en el período 2000-2013, se ha comportado de manera significativamente creciente, tal y como se muestra en la Figura 3-2. Los puntos de la línea roja representan la cantidad de registros identificados en cada año, mientras que la línea negra representa la tendencia al aumento de dicha producción científica.

[4]Complemento de Microsoft Excel
[5]Disponible en: http://www.viscovery.net/somine/
[6]Disponible en: http://www.arschile.cl/ucinet/
[7]Versiones disponibles en: http://www.analytictech.com/netdraw/netdraw_versions.htm

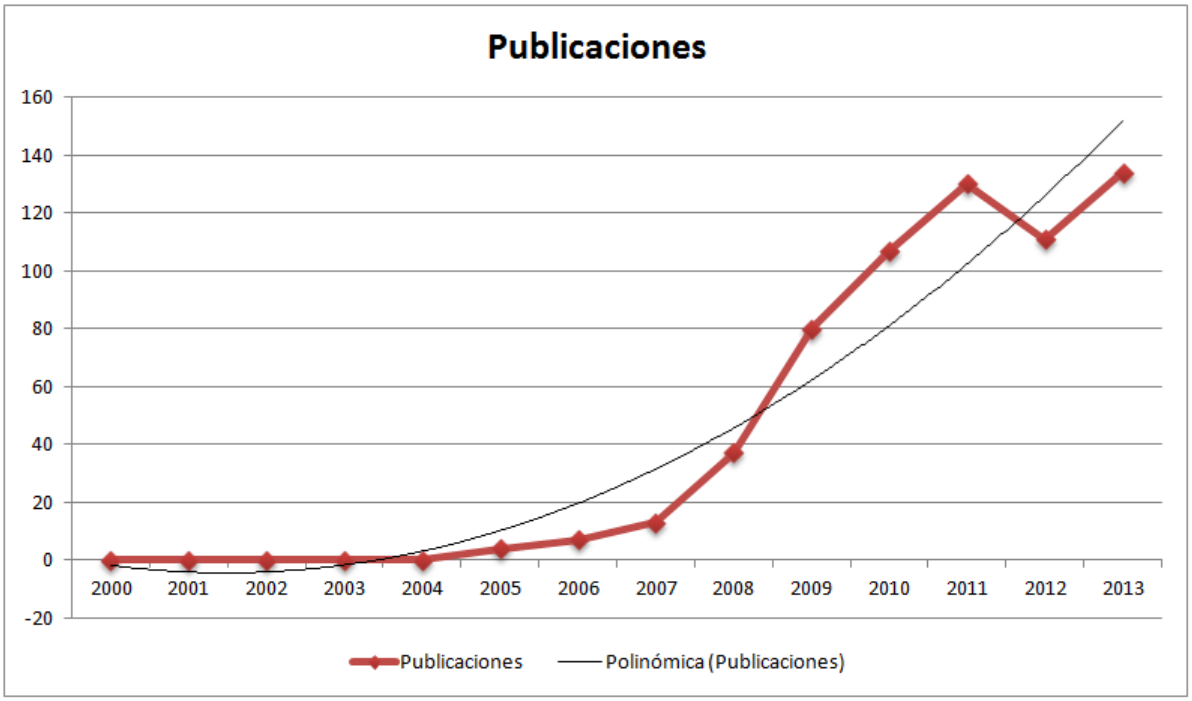


Figura 3-2: Evolución del L2R.

Para el año 2005 se habían identificado solamente 4 registros bibliográficos sobre este tema, sin embargo, en el 2010, la producción científica había aumentado a 107 registros. Una vez terminado el año 2013 se habían recuperado un total de 134 publicaciones.

Esta tendencia apunta indiscutiblemente a que la investigación en el campo del L2R está cobrando interés por la comunidad científica, es un área que va en avance y por tanto, vale la pena dedicar esfuerzos para su investigación.

3.5.2. Temáticas más tratadas o exploradas en L2R

Las temáticas más tratadas son aquellos temas más abordados en los artículos publicados sobre determinada área del conocimiento.

Para realizar el análisis de este indicador se seleccionó el campo "descriptor" y en un primer momento se obtuvieron 3.263 temáticas. Luego, se inicio un procedimiento simple para detectar términos iguales, pero etiquetados de forma diferente. Finalmente, se redujo la lista de temáticas a 3.104 descriptores. De ello resultó que las temáticas más tratadas son: *Information Retrieval* (IR) (288), *data sets* (106), *ranking functions* (100), *search engines* (94), *learning algorithms* (94), *algorithms* (86), *ranking model* (83), y así sucesivamente.

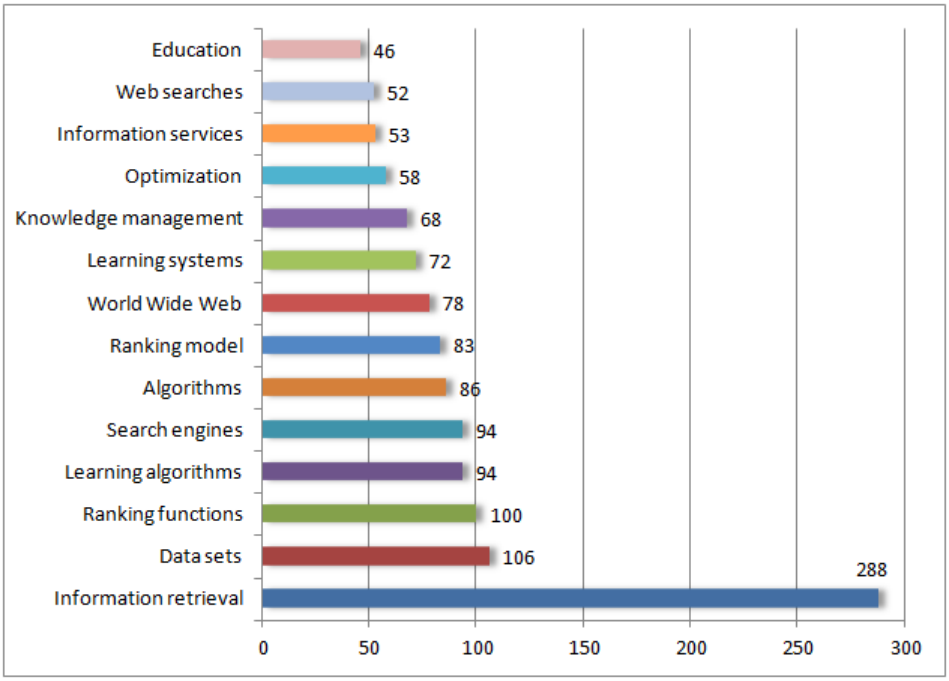


Figura 3-3: Temáticas más exploradas en L2R.

Los temas de mayor significación, y que poseen una frecuencia de aparición superior a 40 registros en el período estudiado, son mostrados en la Figura 3-3. Específicamente en los apartados del 3.6.1 al 3.6.6 de este trabajo, se realiza un análisis detallado y completo de las temáticas tratadas en el L2R.

3.5.3. Áreas del conocimiento en las que más se aplica el L2R

La Figura 3-4 representa las áreas del conocimiento en las que más se aplica el L2R. Como se observa, el área del conocimiento de mayor vinculación es *Computer Science* (ciencia de la computación), con un total de 525 publicaciones para un 55.9 % del total recuperadas. Esto evidentemente está asociado al alto nivel de desarrollo y eficiencia que exigen las tecnologías y tendencias en el mundo de la ciencia de la computación, tanto para el progreso de la ciencia, como para la creación de herramientas y aplicaciones informáticas. El L2R también ha sido aplicado con resultados prometedores en otras esferas de vital importancia como *Mathematics* (matemáticas) con 107 publicaciones, *Decision Sciences* (ciencias de la decisión) con 82, *Engineering* (ingeniería) con 80, *Business, Management and Accounting* (negocio, dirección y contabilidad) con 66, entre otras. El Apéndice A.1

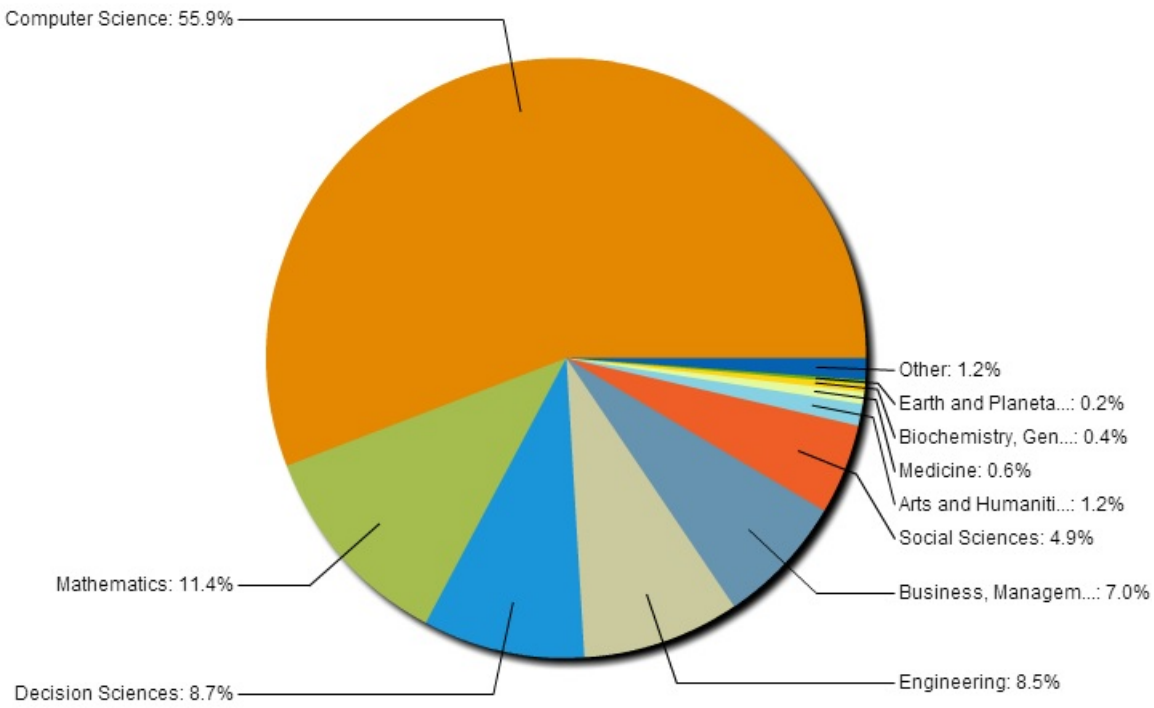


Figura 3-4: Áreas del conocimiento en las que más se aplica el L2R.

muestra otras áreas del conocimiento en las que también se aplica en menor o mayor grado el L2R.

3.5.4. Productividad científica de autores en el L2R

La cantidad de autores identificados que abordan el tema del L2R en mayor o menor grado es de 1.083. El análisis de este indicador arrojó como resultado, que los autores más productivos son Hang Li con 28 publicaciones, Tie Yan Liu con 21, Y. Wang con 17, y así sucesivamente tal y como se muestra en la Figura 3-5. El resto de los autores que han investigado sobre el tema son numerosos, notándose que la gran mayoría de la producción se concentra en un grupo reducido de autores que son los que se muestran en la Figura 3-5. El Apéndice A.2 muestra los nombres y cantidad de publicaciones de los 30 autores más productivos dentro del L2R.

3.5.5. Impacto de la productividad de los autores en el L2R

Un parámetro clave a considerar en el análisis del estado del arte de una determinada disciplina lo constituye el estudio de los trabajos de los autores más referenciados en la

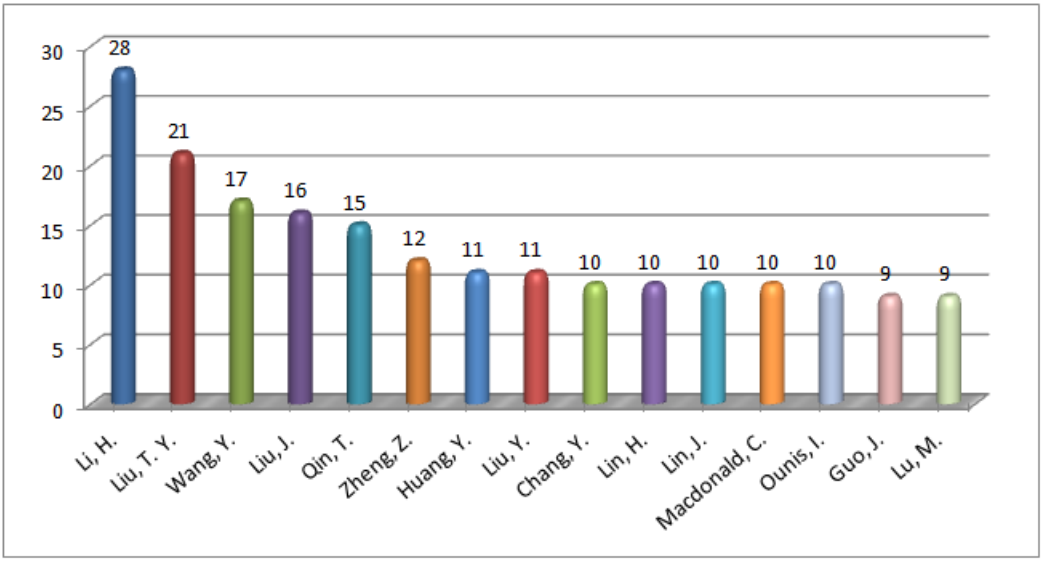


Figura 3-5: Autores de mayor productividad científica en L2R.

literatura científica sobre dicha temática.

Para determinar los resultados del indicador, impacto de la productividad de los autores en el L2R, se realizó el siguiente procedimiento: inicialmente, se obtuvieron todas las referencias bibliográficas de cada uno de los registros recuperados. Posteriormente se contabilizó para el período estudiado la totalidad de citas bibliográficas donde se referenciaba a cada uno de los autores ya conocidos (ver sección anterior).

En la Figura 3-6 se muestran los 20 autores más referenciados en los últimos 10 años, donde sobresalen Thorsten Joachims y Hang Li. Las barras horizontales representan la cantidad de referencias realizadas a sus trabajos.

3.5.6. Fuentes de información más relevantes en L2R

El análisis de este indicador consistió en identificar las fuentes de información más relevantes en L2R según el tipo de referencia.

La distribución de las publicaciones considerando el tipo de referencia resultó ser: en actas de conferencias internacionales (*Conference Proceedings*) 397 publicaciones, artículos en revista (*Journal Article*) 143, en colecciones monográficas (*Book Series*) 81 y en libros (*Books*) sólo 2 publicaciones.

En actas de conferencias internacionales las cinco fuentes de información más relevantes

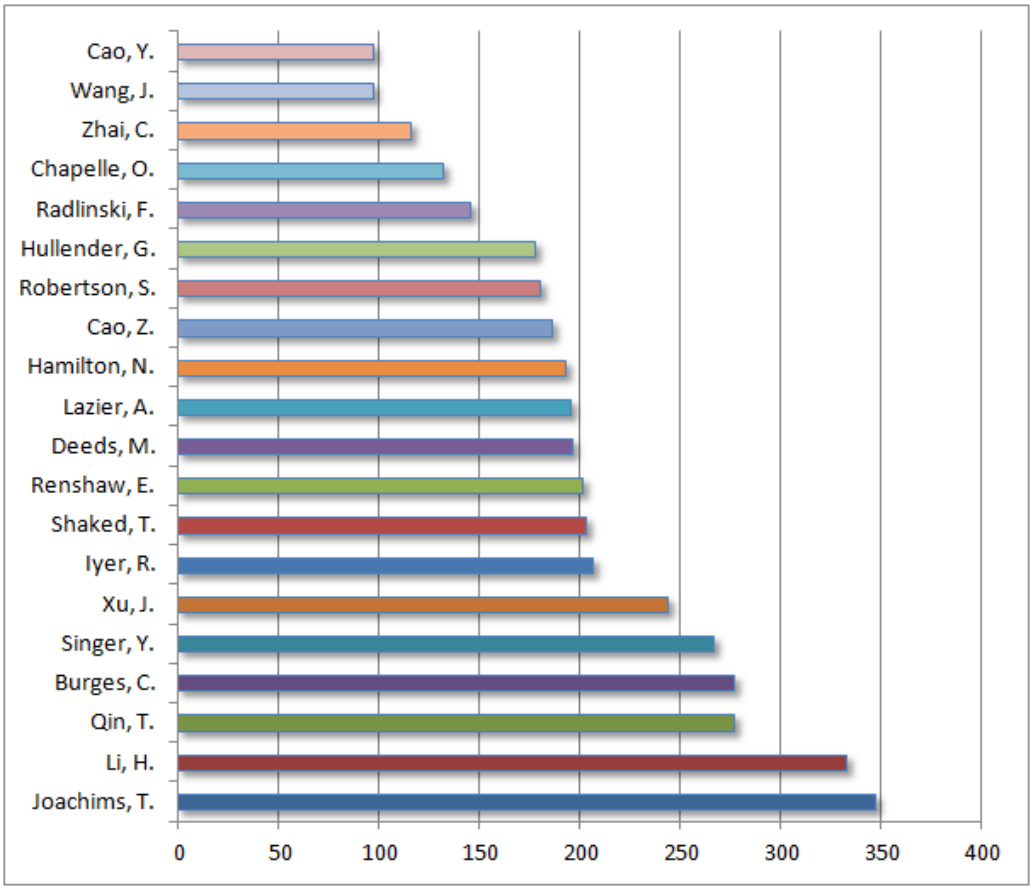


Figura 3-6: Autores más referenciados en los últimos 10 años.

son: *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (84), *ACM International Conference on Information and Knowledge Management* (63), *Annual Conference on World Wide Web* (19), *ACM International Conference on Web Search and Data Mining* (17) y el *Annual International Conference on Machine Learning* (12). El número especificado entre paréntesis representa la cantidad de artículos publicados en la fuente correspondiente. Las publicaciones acumuladas de estas primeras cinco fuentes de información representan el 49.12 % de la totalidad de publicaciones sobre L2R en actas de conferencias internacionales. Donde podemos apreciar además que la conferencia internacional y anual del SIGIR (*Special Interest Group on Information Retrieval*) establece una fuerte frontera de relevancia, por lo que constituye uno de los principales pilares de referencia para conocer lo más novedoso que se viene desarrollando en L2R.

En cuanto a los artículos en revista, las fuentes de información más relevantes son: Information Retrieval (17), Machine Learning (11), Information Processing and Management (7) y Journal of Information and Computational Science (5). Como era de esperar la "Information Retrieval" constituye la revista más predominante y de mayor incidencia productiva por los autores del área.

Finalmente, las dos fuentes de información más relevantes en el ámbito de las publicaciones seriadas son: *European Conference on Information Retrieval* (11) y el *Asia Information Retrieval Societies Conference* (8).

3.5.7. Países con mayor actividad científica en el L2R

El análisis de este indicador permitió conocer los países en los que más se produce información científica sobre L2R. La Figura 3-7 representa, utilizando un gráfico circular, la cantidad de publicaciones correspondientes a cada país, destacándose como los más productivos en el tema, China con 245 publicaciones y Estados Unidos con 172. Le siguen Reino Unido con 47, Canadá con 23 y así sucesivamente. El Apéndice A.3 muestra detalladamente los países y la cantidad de publicaciones generadas por cada uno de ellos.

3.6. Indicadores multidimensionales

El segundo grupo de indicadores denominado relacionales o multidimensionales, de primera y segunda generación, permite conocer las relaciones e interacciones entre elementos de los campos bibliográficos como autor, título, resumen, descriptores o palabras clave y referencias bibliográficas. Entre los indicadores bibliométricos multidimensionales más utilizados se pueden mencionar: temáticas más tratadas por años, temáticas más tratadas por los autores más productivos, temáticas más tratadas en las fuentes de información más relevantes, comportamiento de la productividad por años y colaboración científica.

3.6.1. Desglose grupal de las temáticas tratadas en L2R

El análisis de los indicadores unidimensionales permitió conocer de manera muy general las temáticas más tratadas sobre L2R (ver Sección 3.5.2).

Si consideramos el cúmulo de descriptores obtenidos (un total de 3.104), resulta necesario

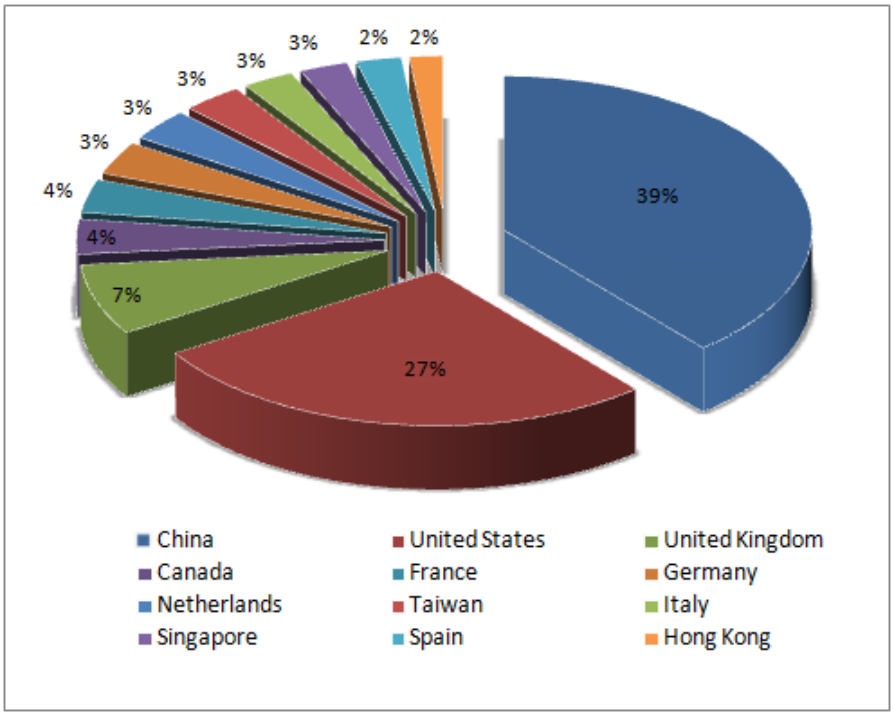


Figura 3-7: Países con mayor productividad científica en L2R.

realizar un estudio detallado para conocer el comportamiento, así como las tendencias en el tratamiento de dichas temáticas en el período analizado.

A fin de llevar a cabo este análisis, los autores del presente trabajo formaron 4 grandes grupos:

- **Grupo 1. Temáticas en auge y estables**: son aquellas que además de ser las más abordadas en los artículos publicados, tienen que cumplir la condición de haber sido tratadas continuamente en al menos cada uno de los últimos cinco años del período estudiado.

- **Grupo 2. Temáticas incipientes-prometedoras**: son aquellas que han sido abordadas por los investigadores del área de forma continuada en cada uno de los últimos 3 años. Estas temáticas pueden en un momento determinado pasar a formar parte del Grupo 1.

- **Grupo 3. Temáticas incipientes-puntuales**: son aquellas que han sido tratadas en uno u otro año reciente, pero que no se percibe en el transcurso de los años una

continuidad en su desarrollo.

- **Grupo 4. Temáticas obsoletas o abandonadas**: son aquellas que tienen asociada una sola publicación y hace cinco o más años que no se tratan o no se tiene constancia de que se hayan tratado.

3.6.2. Temáticas en auge y estables

Analizando toda la muestra y las condiciones establecidas para pertenecer al Grupo 1, se pudo constatar una vez finalizado el año 2013 que un total de 34 temáticas podían ser consideradas en auge y estables.

En los mapas topológicos correspondientes a las Figuras 3-8 y 3-9, podemos apreciar la intensidad, el agrupamiento y las relaciones (fronteras) existentes entre cada una de las temáticas, considerando la producción científica relacionada con cada temática en particular, para los años 2009 y 2013, respectivamente.

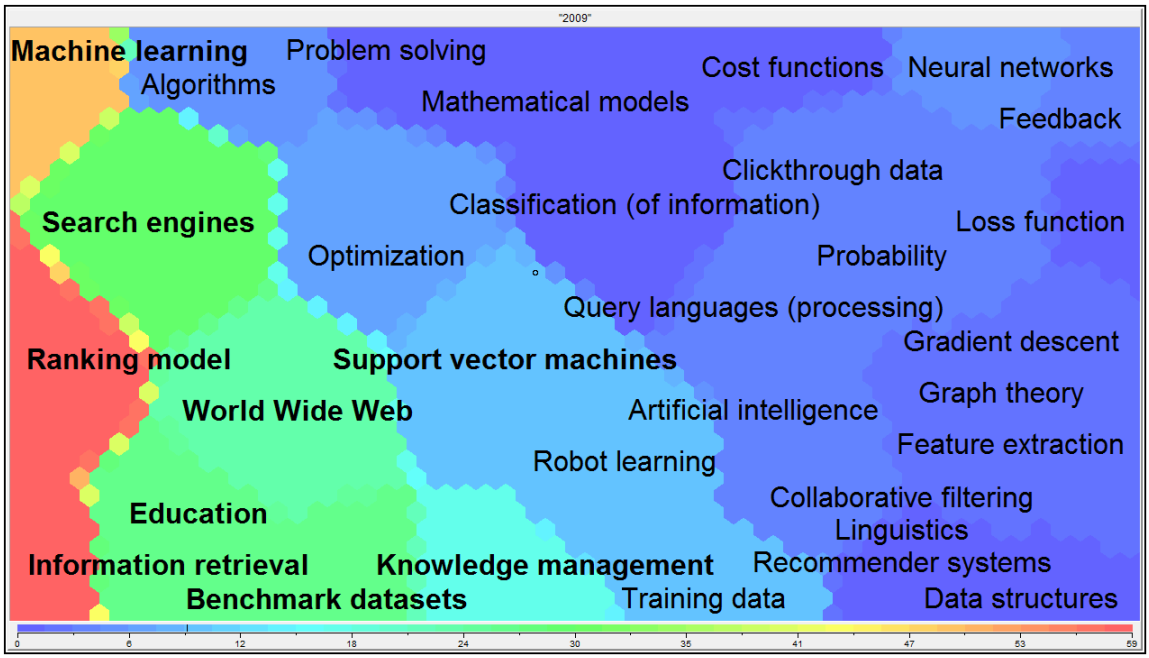


Figura 3-8: Temáticas en auge y estables en el año 2009.

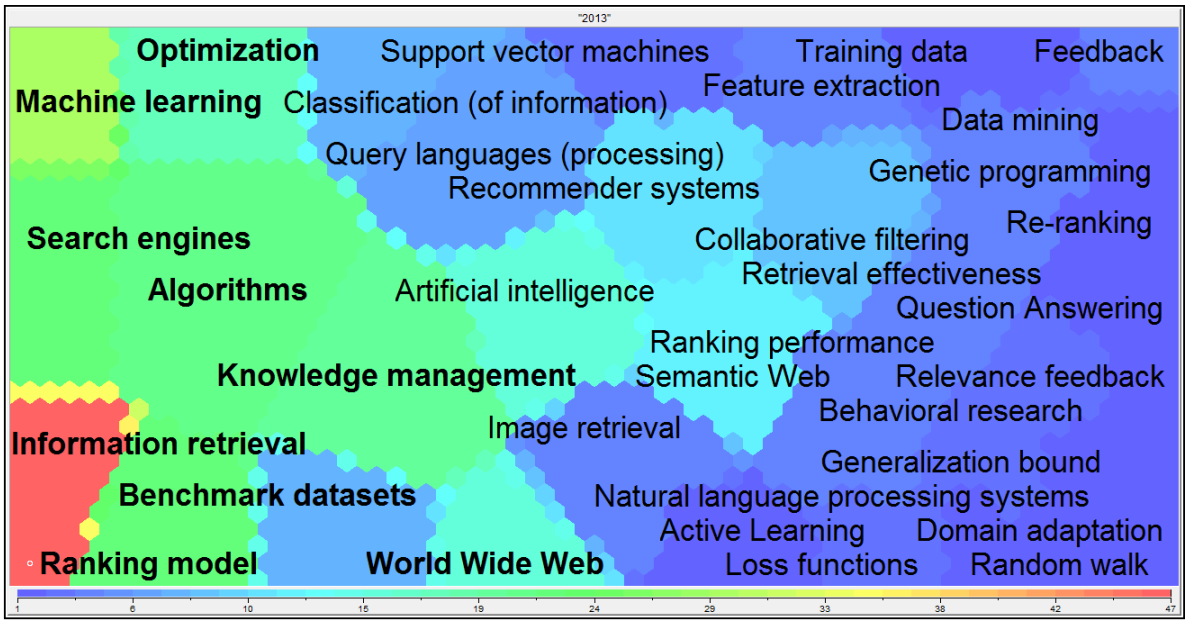


Figura 3-9: Temáticas en auge y estables en el año 2013.

Para ilustrar este estudio, se seleccionaron los años 2009 y 2013. El 2009 porque refleja desde el año 2005 donde comienza a incrementarse la producción científica sobre el L2R cuáles fueron aquellas temáticas de mayor interés para los primeros autores del área. En el caso del 2013 es evidente, pues representan la tendencia y la evolución de las temática más tratadas sobre L2R, desde hace cinco años hasta la actualidad.

Específicamente, la intensidad de la producción científica de cada temática está representada en cada mapa por una tonalidad de colores (desde azul hasta rojo) que, según una escala numérica correspondiente, representa la cantidad de publicaciones que para ese año tuvo una determinada temática (según el color del cluster donde esté situada). Por ejemplo, podemos decir que para el año 2009, se publicaron aproximadamente un total de 71 artículos sobre *Information Retrieval*; sin embargo en el caso de la temática *Collaborative filtering* (filtrado colaborativo) no superaba las 3 publicaciones para este año.

A pesar de que podemos ver que muchas temáticas tienen la misma intensidad de publicaciones, estás no están concentradas en una misma zona; sus clústeres están ubicados de tal forma que a través de sus fronteras pueda ser apreciada además su relación y proximidad con otras temáticas. Por ejemplo, en el año 2013, a pesar de que las temáticas *Artificial intelligence* y *Training data*, tienen igual rango de intensidad, por la posición de sus

cluster, podemos concluir que "*Artificial intelligence*" es una temática más prometedora pues tiene fronteras con *Search engines*, *Algoritms*, entre otras temáticas que estuvieron liderando la producción en ese año.

Es importante mencionar que si analizamos concretamente estos dos mapas topológicos, además de notar como, con el paso de los años, entran y salen de escena varias temáticas, existe un grupo de temáticas que ha sido estable en su aparición y en la relación que establecen entre ellas. Entre estas últimas, resaltan las relaciones y la intensidad de *Information Retrieval*, *Ranking model*, *Search engines* y *Machine learning*.

Con la intención de enriquecer mucho más la información que brindan los mapas de las Figuras 3-8 y 3-9, y llegar a una mejor interpretación del comportamiento de estas temáticas en auge y estables, estas fueron etiquetadas de la siguiente manera: las temáticas de color negro sin enfatizar representan aquellas en las que la producción de artículos acumulados en todos los años comprendidos hasta ese instante de tiempo fue menor que la media aritmética considerando la cantidad de publicaciones por temática. Por su parte, las enfatizadas en color negro corresponden con producciones superiores a este valor promedio.

Podemos decir entonces, que la temática más tratada una vez finalizado el año 2013 fue *Information retrieval* con un total de 336 artículos asociados. Las temáticas que más prevalecen con una cantidad superior a 74 publicaciones son: *ranking model*, *knowledge management* (gestión del conocimiento), *search engines* (motores de búsqueda), *algorithms* (algoritmos), *optimization* (optimización), *machine learning* (aprendizaje automático), *Benchmark datasets* (conjuntos estándares de datos), y *World Wide Web*.

La proximidad entre los clústeres en los que se enmarca cada temática nos muestra además, la relación existente entre cada una de ellas en particular, así como la fuerte vinculación entre las más tratadas o exploradas en cada año. Por lo cual, también podemos concluir que entre estas temáticas relacionadas existe un fuerte lazo y que en menor medida, pero con una fuerte vinculación, se abordaron otros temas como: *support vector machines* (máquinas de soporte vectorial), *Query languages (processing)* (lenguajes de consulta(tratamiento)), *Recommender systems* (sistemas de recomendación), *natural language processing systems* (sistemas de procesamiento del lenguaje natural), y así sucesivamente podemos seguir asociando otras temáticas.

3.6.3. Temáticas incipientes

Las temáticas incipientes pueden ser catalogadas como aquellas que de una forma u otra emergen como resultado de nuevas investigaciones y trabajos. En sentido general, la mayoría de las temáticas que surgen no llegan a imponerse como puntos de atención dentro de la comunidad científica del área que se investiga en particular, pues tienen un impacto limitado. Estas temáticas que son tratadas en uno u otro momento, pero que no se percibe una continuidad en su desarrollo, la consideramos como temáticas incipientes-puntuales (temáticas del Grupo 3).

Por otro lado, tenemos las temáticas del Grupo 2, o temáticas incipientes-prometedoras donde se enmarcan aquellas temáticas que han sido tratadas continuamente en cada uno de los últimos 3 años. Es importante, estar al tanto del comportamiento de tales temáticas, pues son las que potencialmente pueden convertirse en un futuro cercano en temáticas del Grupo 1 (en auge y estables).

La Figura 3-10 nos muestra un esquema general de las temáticas clasificadas como incipientes dentro del área L2R según los criterios establecidos por los Grupos 2 y 3.

Para una mejor interpretación de las 64 temáticas incipientes obtenidas, el esquema fue estructurado en 6 cuadrantes representativos. Estos cuadrantes son formados por dos columnas verticales que separan indistintamente las temáticas incipientes-prometedoras (2011-2013) de las temáticas incipientes-puntuales (2009-2013). Desde el punto de vista horizontal, se establecieron 3 filas, la primera y ubicada en la posición más inferior, enmarca las temáticas de L2R que en su tratamiento han generado hasta 5 publicaciones científicas. La fila segunda e intermedia abarca temáticas en las que se ha trabajado un poco más, es decir, de 6 a 10 publicaciones. Por su parte, la última fila agrupa aquellas temáticas que han propiciado una producción científica de más de 10 artículos.

Siguiendo estas divisiones, y a partir de la información obtenida en el procesamiento, podemos identificar fácilmente a *computational linguistics* (lingüística computacional) como el tema incipiente-prometedor más continuamente tratado en los años 2010, 2011, 2012 y 2013. También se han manifestado con mayor o menor grado otras temáticas que pudiesen ser prometedoras. A continuación se listan las más sugestivas: *transductive learning* (aprendizaje transductivo), *semantic web* (web Semántica), *document ranking*, *persona-*

lization, *crowdsourcing* (colaboración abierta distribuida) y así sucesivamente podemos prioritariamente enumerarlas todas. Por otro lado, sería interesante dirigir la atención a las

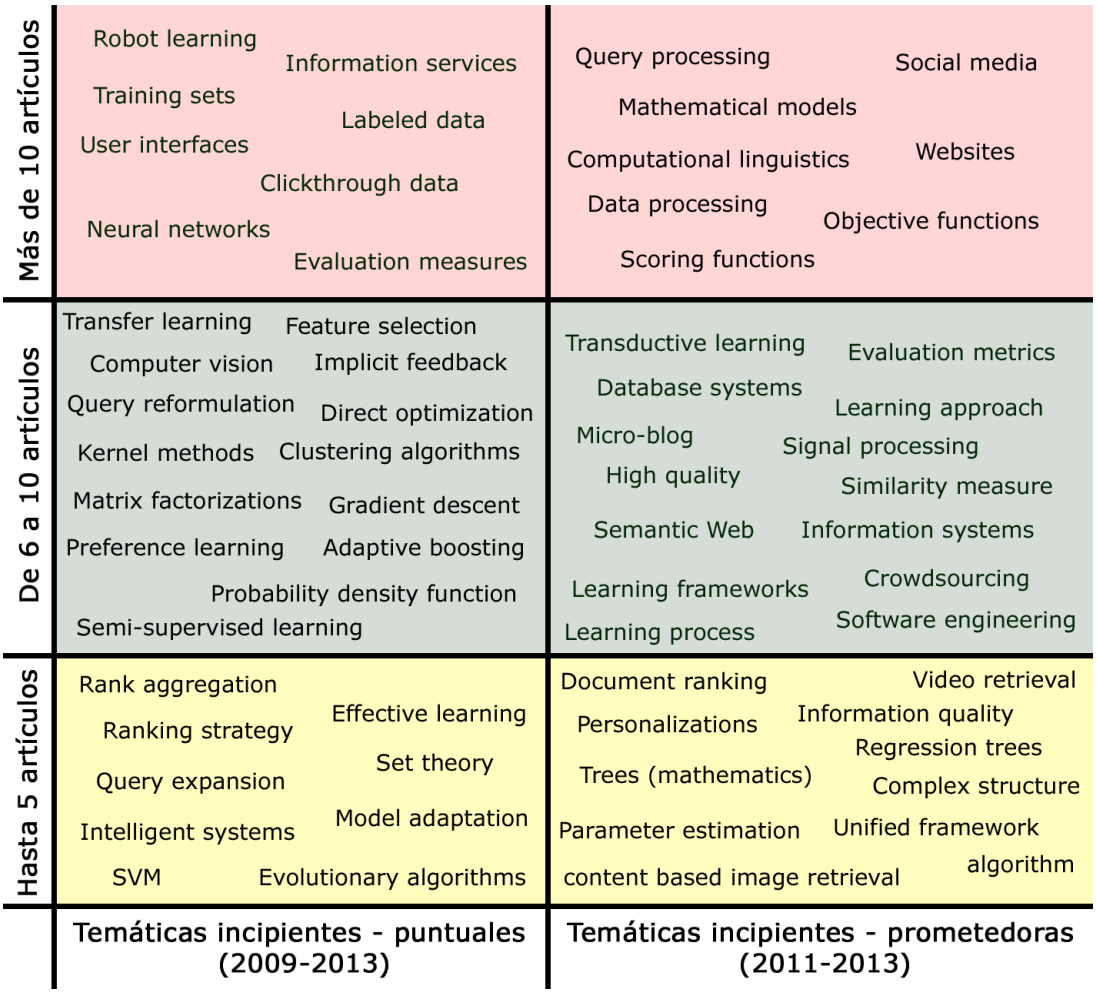


Figura 3-10: Esquema de temáticas incipientes en L2R.

siguientes temáticas puntualmente tratadas: *evaluation measures* (medidas de evaluación), *robot learning*, *adaptive boosting*, *semi-supervised learning* (aprendizaje semi-supervisado), *feature selection* (selección de rasgos), *evolutionary algorithms* (algoritmos evolutivos) y *rank aggregation* (agregación de ordenamientos).

Es importante mencionar que en la versión 4.0 de LETOR[8], liberada en julio de 2009, una de las tareas del L2R está encaminada y relacionada directamente con las temáticas: *semi-supervised ranking* y *rank aggregation*.

[8]Disponible en: http://research.microsoft.com/en-us/um/beijing/projects/letor/

3.6.4. Temáticas obsoletas o abandonadas

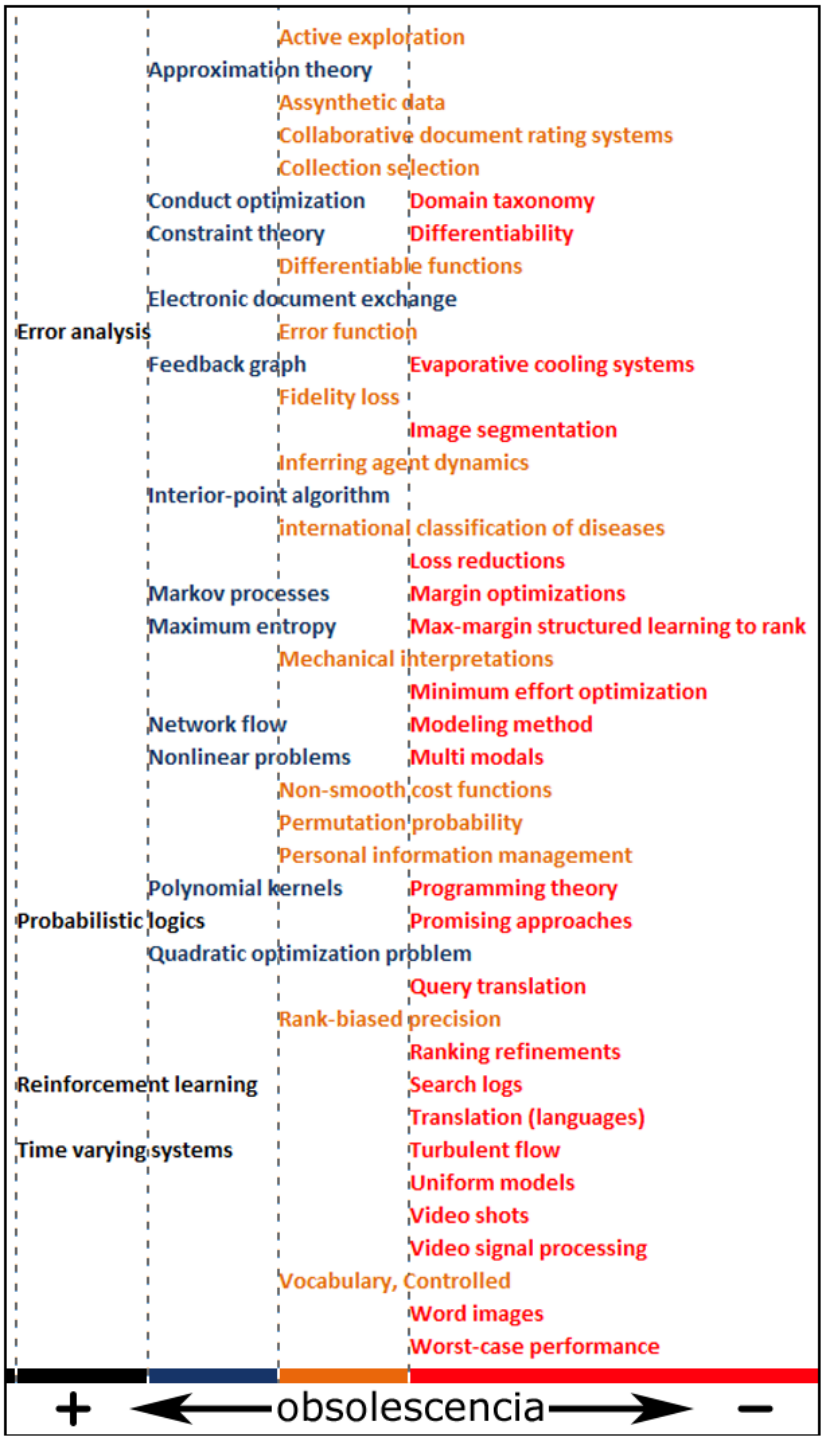


Figura 3-11: Esquema simbólico que representa la tendencia a la obsolescencia en el tratamiento de algunas temáticas en L2R.

En el análisis realizado se determinó que existe un gran número de temáticas que envejecen o quedan abandonadas en el tiempo en comparación con otras que emergen (antes mencionadas).

Estas temáticas pertenecen según el agrupamiento realizado en este trabajo al Grupo 4, denominado temáticas obsoletas o abandonadas.

Todas estas temáticas tienen asociadas el menor índice de publicaciones (un sólo artículo), el cual fue publicado hace cinco o más años atrás.

En el balance realizado finalmente se determinó que 53 temáticas pueden ser consideradas obsoletas o abandonadas.

En la Figura 3-11, podemos observar un esquema simbólico que representa la tendencia a la obsolescencia en el tratamiento de algunas temáticas en el L2R.

Con el objetivo de ilustrar en detalle el comportamiento de tales temáticas, en el esquema se hace uso de un etiquetado con diferentes colores. Las temáticas de color rojo simbolizan aquellas que hace 6 años no se tratan o por lo menos no se tienen constancia de publicaciones en ese período, las de color naranja referencian 7 años atrás, las azules 8 años y las negras 9 años.

Los símbolos (+) y (-), conjuntamente con la orientación de las flechas, dan a entender la tendencia de cada temática (según su ubicación) a convertirse en más o menos obsoleta.

Error analysis (análisis del error), *probabilistic logics* (lógica probabilística), *reinforcement learning* (aprendizaje basado en el refuerzo) y *time varying systems* (sistemas variables en el tiempo), constituyen las temáticas más obsoletas o abandonadas en el L2R.

3.6.5. Temáticas más tratadas por los autores de mayor impacto

Analizando las temáticas desde otro punto de vista, sería interesante determinar qué temáticas son las más tratadas por los autores de mayor impacto. Con este fin, se seleccionaron los 50 autores de mayor impacto y se tabularon las correspondencias de los trabajos de estos autores con respecto a los registros donde se tratan dichas temáticas.

En la Tabla 3-1, se muestran las 20 temáticas más tratadas por los autores de mayor impacto. La columna etiquetada como “Total” representa la sumatoria de la cantidad de

veces que estos autores de impacto abordan cada temática en todos sus trabajos.

Tabla 3-1: Temáticas más tratadas por los 50 autores de mayor impacto.

Orden	**Temáticas**	**Total**
1	Information retrieval	77
2	Learning systems	52
3	Learning algorithms	45
4	Ranking functions	40
5	Ranking model	37
6	Support vector machines	34
7	Data sets	33
8	Information services	31
9	Optimization	30
10	Education	29
11	Algorithms	25
12	Search engines	25
13	Information retrieval systems	21
14	Machine learning	20
15	Loss functions	19
16	Document retrieval	18
17	Problem solving	18
18	World Wide Web	18
19	Database systems	16
20	Knowledge management	16

Finalmente, los resultados obtenidos corroboran que la mayoría de las temáticas en auge y estables (ver apartado 3.6.2) son tratadas por los autores de mayor impacto.

Como información adicional, podemos mencionar que Hang Li (78), T. Qin (47), H. Zha (41), Y. Huang (35), Z. Zheng (31), G. Sun (26), J. Xu (26), Y. Chang (20), D. Metzler (19) y J. Lin (18), son los 10 autores de impacto que más han tratado las temáticas obtenidas. El número especificado entre paréntesis representa la sumatoria de la cantidad de veces que cada autor trata estas 20 temáticas.

3.6.6. Temáticas más tratadas en las fuentes de información más relevantes

El análisis de este indicador consistió en identificar las temáticas más tratadas en las fuentes de información más relevantes, es decir, aquellas fuentes que más publican sobre las temáticas relacionadas con el tema "*Learning to rank*".

Según la descripción de la Sección 3.5.6. (Fuentes de información más relevantes en L2R), en actas de conferencias internacionales el *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* se establece como la fuente de información más relevante con 84 publicaciones. En cuanto a los artículos en revista la fuente más relevante es *Information Retrieval* con un total de 17 publicaciones.

En el mapa de la Figura 3-12 se visualizan las temáticas más tratadas en la fuente de información *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Las temáticas más sobresalientes son: *Information Retrieval* (IR), *search engines*, *information services* (servicios de información), *ranking functions*, y así sucesivamente. Resulta notable reconocer que se han publicado aproximadamente 68 artículos donde se trata de alguna forma la RI. Además, a pesar de que unas temáticas son mucho más tratadas que otras, como por ejemplo, *natural language processing systems* y *ranking functions*; se establece entre ambas una estrecha relación (dada por los autores) por la proximidad de sus fronteras.

Por otra parte, las temáticas más tratadas en las principales publicaciones de las revistas más relevantes son: *Information retrieval* (IR), *learning algorithms*, *data sets*, *ranking functions*, entre otras. En la Figura 3-13, puede ser visualizada en detalle la relevancia y relación existente entre cada una de las temáticas. Por demás, en este mapa topológico se evidencia, como la distribución y la relación entre las temáticas está totalmente condicionada a la intensidad de aparición de cada una de estas; es decir, las temáticas mas tratadas se relacionan con aquellas un tanto menos tratadas, pero no con aquellas mucho menos tratadas. Por ejemplo, *Information Retrieval* (IR) se relaciona con *learning algorithms* y *data sets*, y así consecutivamente, pero no tiene fronteras intermedias con *neural networks* u otra temática menos tratada.

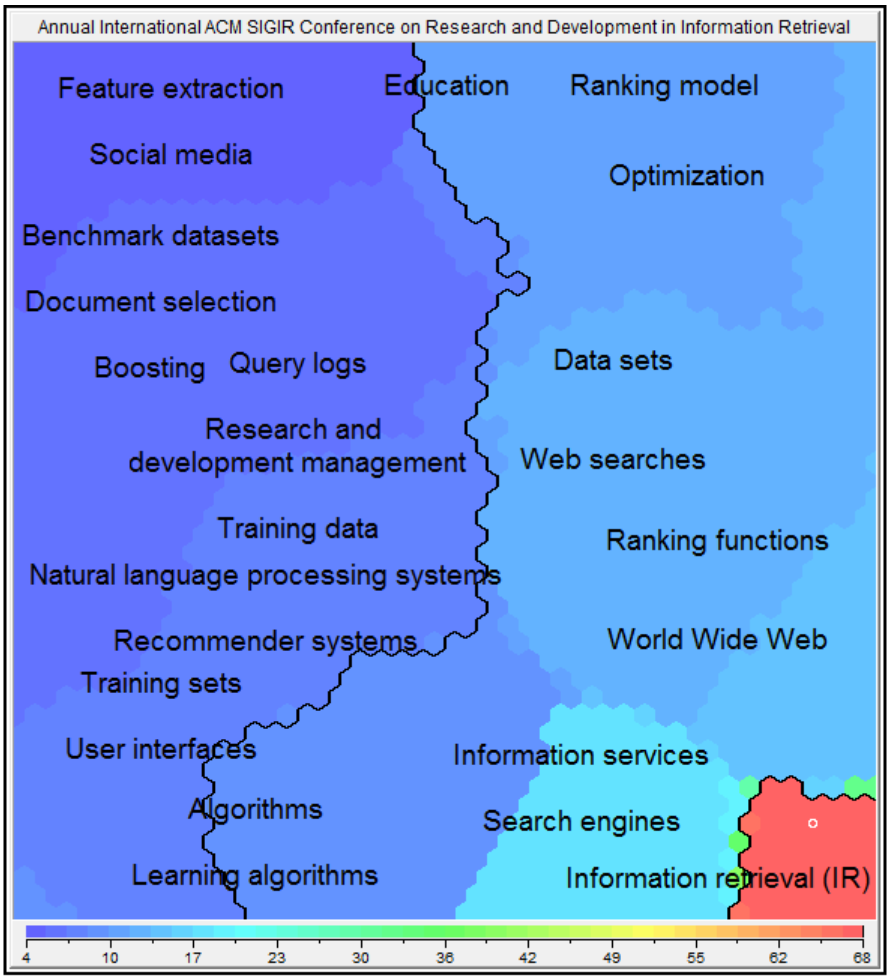


Figura 3-12: Temáticas más tratadas en la Conferencia Internacional Anual del SIGIR.

De manera general, según la distribución y la intensidad que tienen cada una de las temáticas en ambos mapas topológicos, podemos concluir que la mayoría de las temáticas más tratadas en las fuentes de información más relevantes, coinciden también con las principales temáticas en auge y estables. Además, podemos apreciar como la cantidad de publicaciones relacionadas con las temáticas más tratadas en la Conferencia Internacional Anual del SIGIR superan significativamente la aparición de estas y otras temáticas publicadas en las revistas más relevantes. Lo cual nos sugiere que el SIGIR, constituye un espacio confiable, prestigioso y actualizado donde concurren y se publican investigaciones de primer orden sobre el L2R. Podemos afirmar también, que en el caso de las revistas más relevantes la relación entre las temáticas más tratadas está bien definida.

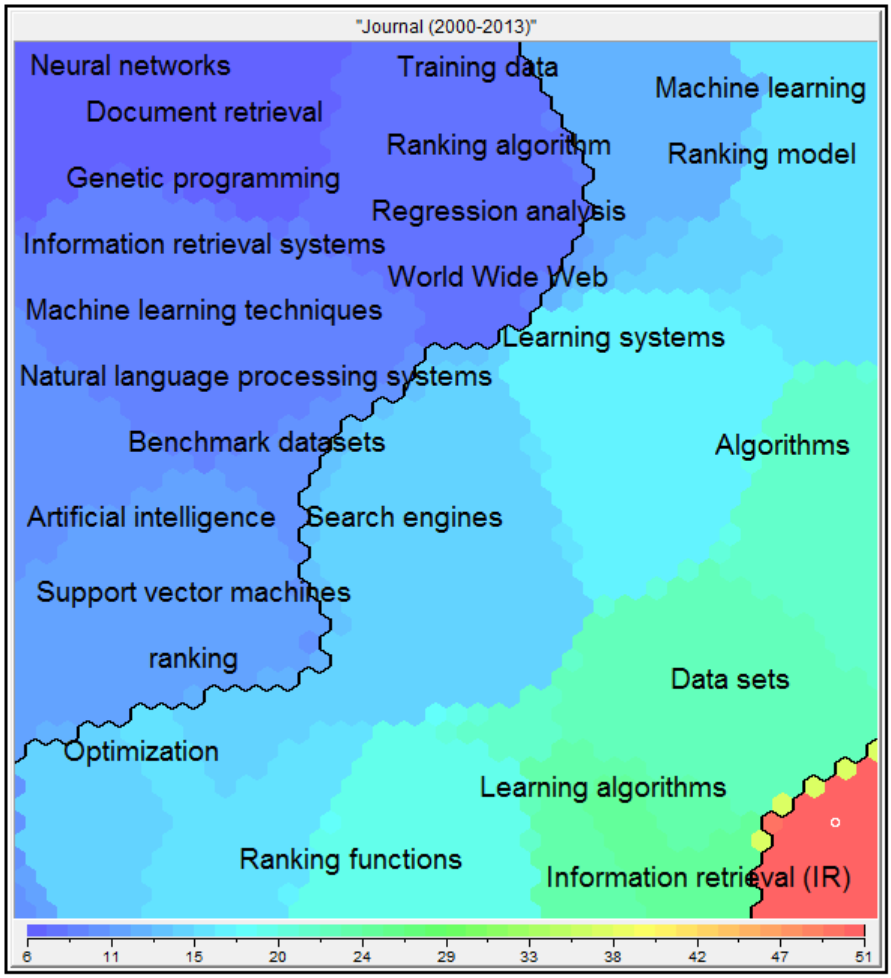


Figura 3-13: Temáticas más tratadas en las revistas más relevantes.

3.6.7. Colaboración de los autores que presentan más de 5 trabajos

Para el análisis de este indicador, se elaboró un grafo para representar el nivel de coautorías a través de una red social de colaboración entre los autores más productivos. Específicamente se consideraron aquellos autores que presentan más de 5 trabajos. Una vez establecidas las relaciones, se eliminaron los autores independientes quedando una red compuesta por enlaces entre 51 autores.

Finalmente como se puede apreciar en la Figura 3-14 se obtuvo una red densa compuesta por 26 autores[9]. Este gran clúster de autores en su mayoría de procedencia China, presen-

[9]Además de este gran clúster, fueron obtenidos también, dos dúos, un trio, un cuarteto, un sexteto y un octeto; lo cual evidencia el trabajo de forma independiente entre otros grupos de autores. Los grafos

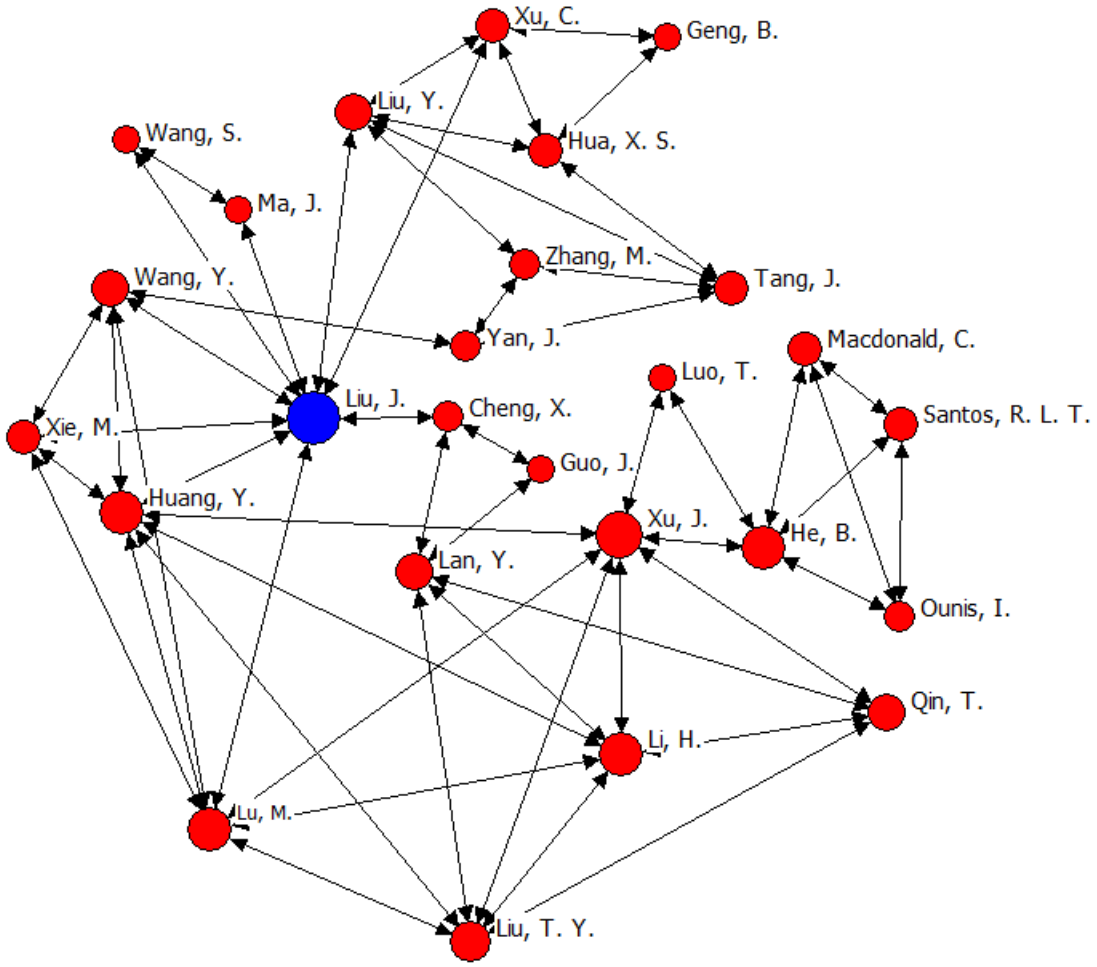


Figura 3-14: Red de colaboración autoral en L2R.

tan una significativa intensidad en sus relaciones. El tamaño de los círculos representa la tendencia de estos autores a colaborar; mientras mayor sean, más colaborativos son dichos autores. Como se puede observar en el grafo de la Figura 3-14, los autores que más tienden a colaborar con otros autores son: J. Liu, Y. Huang, M. Lu, J. Xu y Hang Li, entre otros, que se visualizan en círculos rojos menos grandes.

3.7. Fundamentos para nuestra investigación

El presente estudio bibliométrico corrobora que el L2R es un área que actualmente muestra una tendencia creciente a la investigación. Este interés de la comunidad científica está dado por la importancia teórica y el impacto de relevancia práctica que tiene el L2R dentro de

de estos grupúsculos de autores pueden ser consultados detalladamente en el Apéndice A.4.

muchas aplicaciones relacionadas con la RI, el procesamiento del lenguaje natural y la minería de datos. Con lo cual es una línea investigativa vigente, en auge y en la que resulta motivador incursionar en nuevas investigaciones.

A partir del conocimiento que se extrae del análisis de cada uno de estos indicadores unidimensionales y multidimensionales, pueden ser planteadas disímiles de vertientes de trabajo enfocadas hacia temáticas específicas considerando su evolución y las perspectivas que cada investigador valore según el grupo categórico en el que se encuentra, o la posible relación con otras ciencias.

A nosotros específicamente nos pareció interesante y oportuno investigar directamente en temáticas como: recuperación de información, funciones de *ranking*, métodos de *ranking*, algoritmos de aprendizaje, algoritmos de *ranking*, modelos de *ranking* y optimización, por ser tópicos en los que se está trabajando con mucha intensidad y en los cuales todavía falta mucho por hacer. Por otro lado, también nos pareció atractivo trabajar de forma innovadora en las temáticas: conjunto de datos y selección de rasgos, esta última una temática incipiente que ha sido tratada puntualmente (no más de 10 artículos) desde el año 2009 hasta finales del 2013. También y para facilitar un tanto nuestra labor investigativa y pensando en futuros desarrollos, finalmente nos planteamos abordar las temáticas herramientas de experimentación y herramienta de análisis de datos, que también han sido tratadas puntualmente y bien dirigidas pudiesen no sólo asistir y organizar nuestro trabajo en etapas de experimentación y análisis, sino además agilizar la obtención de resultados y con ello la toma de decisiones oportuna.

Los resultados obtenidos de estas investigaciones son descritos en los siguientes capítulos de esta memoria documental.

3.8. Conclusiones

Entre los principales resultados del estudio métrico, se puede mencionar como el más relevante, que la investigación científica sobre los temas de L2R en los últimos 5 años, ha crecido significativamente. Otro de los aportes importantes del trabajo, está dado por la clasificación y agrupamiento de todas las temáticas relacionadas con el L2R. Esta representación categórica dejó claro que existen temáticas como por ejemplo *information*

retrieval, *ranking model* y *machine learning*, que desde los inicios del L2R han sido tratadas continuamente por los principales autores del área. Mientras que temáticas como *feature selection* y *rank aggregation* han sido tratadas puntualmente a lo largo del tiempo, y otras como *reinforcement learning* hace más de 8 años que no son abordadas.

En todo este análisis también se pudo corroborar que la Conferencia Internacional Anual del SIGIR establece una fuerte frontera de relevancia en cuanto a las principales temáticas que se están tratando en cada momento histórico, por lo cual constituye uno de los principales pilares de referencia para conocer lo más novedoso que se viene desarrollando en L2R.

Finalmente, se puede afirmar que en esta área existe una fuerte tendencia colaborativa entre los principales autores, donde sobresalen J. Liu, Y. Huang, M. Lu, J. Xu y Hang Li.

PARTE III

Modelos y estrategias para Learning to Rank

Capítulo 4

Aprendizaje de la Ordenación basado en Optimización con Enjambre de Partículas

"La ciencia es la progresiva aproximación del hombre al mundo real".
— Max Planck

4.1. Introducción

En el presente capítulo proponemos un nuevo método de Aprendizaje de la Ordenación, denominado RankPSO. Este algoritmo de aprendizaje, basado en la Optimización con Enjambre de Partículas (PSO), construye un modelo de ordenación capaz de optimizar directamente cualquier medida de rendimiento empleada en la Recuperación de Información. Implementa además una estrategia para evadir la convergencia temprana hacia mínimos locales. Se detalla también un procedimiento de dos pasos para reducir la dimensionalidad de los datos en *datasets* estándares en L2R. Este procedimiento fue concebido combinando los métodos Análisis de Componentes Principales (PCA) y análisis de clúster. Se llevó a cabo un estudio experimental utilizando todos los *datasets* de la colección de datos ".Gov" para evaluar el rendimiento de RankPSO y su efectividad en comparación con otros métodos referenciados en la literatura. Los resultados obtenidos fueron analizados estadísticamente, y finalmente se describe un análisis de la influencia del procedimiento de reducción de dimensionalidad en relación al tiempo de respuesta y al rendimiento en la ordenación.

4.2. Optimización con enjambre de partículas

En las siguientes dos secciones presentaremos el método clásico de PSO, su formulación y parámetros claves, y haremos alusión a la incorporación del peso inercial como elemento importante en la convergencia de este método.

4.2.1. PSO Clásico

La Optimización con Enjambre de Partículas[1] (PSO, por sus siglas en inglés), es una técnica de optimización estocástica basada en poblaciones, inspirada en el comportamiento social observado en animales o insectos (por ejemplo, bancos de peces, bandadas de aves, etc.) [93]. Fue propuesta por primera vez en 1995 por Kennedy y Eberhart [94].

Básicamente, los individuos en PSO reciben el nombre de partículas y cada partícula i se compone de un vector de posición x_i (coordenadas en el espacio de búsqueda), un vector de velocidad v_i, que define el desplazamiento de esa posición, y una memoria de la mejor solución encontrada hasta el momento por ella, p_i. En particular, la velocidad de una partícula está determinada tanto por p_i como por la memoria global del enjambre g_{best} (la mejor entre todas las partículas). Existen modelos que utilizan la mejor solución de una determinada vecindad, es decir, considerando sólo una parte del enjambre. A estos modelos se les conocen como l_{best}, a diferencia del habitual g_{best}, que es el considerado en la presente investigación. La diferencia estriba en que en l_{best}, la memoria g_{best} es sustituida por el mejor localmente, l_{best} [95].

Durante la evolución del enjambre, cada partícula i actualiza su valor de velocidad y posición con respecto a la iteración anterior, a partir de las siguientes ecuaciones [96]:

$$v_i^{(t+1)} = v_i^{(t)} + c_1 n_1 \circ \left(p_i - x_i^{(t)}\right) + c_2 n_2 \circ \left(g_{best} - x_i^{(t)}\right); \tag{4-1}$$

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}; \tag{4-2}$$

donde el súper índice (t) hace alusión al valor de velocidad y posición que traía la partícula de la iteración anterior, y $(t+1)$ referencia la nueva velocidad y posición que tomará para esta iteración actual. Los n_1 y n_2 son vectores n-dimensionales formados por números aleatorios en el rango [0,1]. Por su parte, las constantes de aceleración c_1 y c_2 representan la ponderación de los términos estocásticos de aceleración que tiran de cada partícula hacia las posiciones p_i y g_{best}. Los valores bajos permiten que las partículas se muevan con cierta soltura lejos de las regiones de destino antes de ser atraidas hacia estas; mientras que los valores altos dan lugar a movimientos abruptos hacia, o más allá de, las regiones de

[1]Traducción literal del término inglés "Particle Swarm Optimization (PSO)".

destino. El operador "$\circ$" especifica un producto Hadamard[2] entre las matrices formadas por las coordenadas de los vectores, es decir, elemento a elemento.

La expresión de la velocidad (ecuación 4-1) en sí engloba el aporte principal de PSO que le permite clasificarse como un paradigma de la Inteligencia de Enjambre [93]. Cada uno de sus componentes tiene un significado relacionado con el comportamiento social de los enjambres en la naturaleza. La primera parte de la ecuación (4-1) es el *momentum*: la velocidad previa que es usada para direccional al individuo en su trayectoria, permitiéndole con un grado de capacidad de memoria la exploración de nuevos espacios de búsqueda. La segunda parte representa el componente cognitivo[3] y la tercera parte es el componente social[4] que manifiesta la colaboración entre las partículas. Entonces, las partículas se mueven hacia nuevas posiciones según la ecuación (4-2). La Figura 4-1 ilustra cómo se desarrolla el movimiento de una partícula en PSO y la influencia de sus componentes.

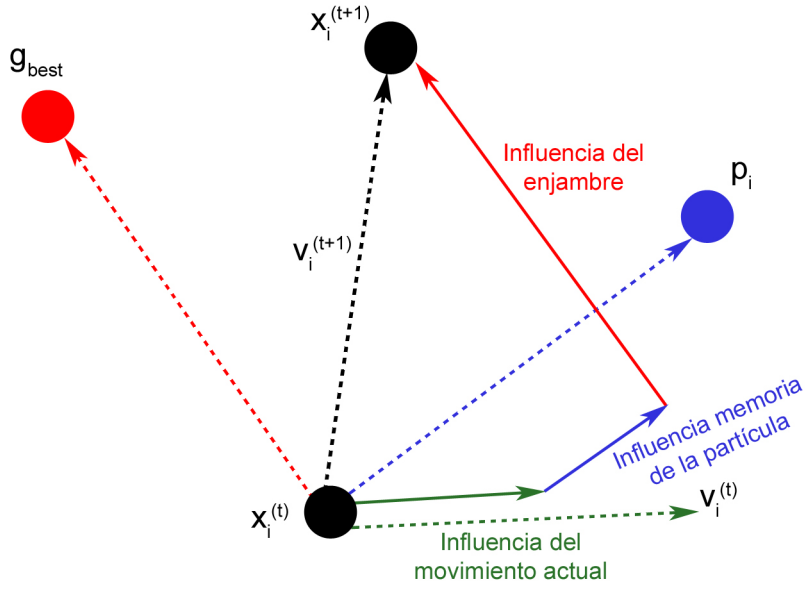


Figura 4-1: Movimiento que experimenta una partícula en PSO - La nueva posición ubica a la partícula en una zona intermedia entre las memorias personal y global.

Durante todo este proceso, el rendimiento de cada partícula se mide según una función de aptitud predefinida.

[2]Sean dos matrices A, $B \in \Re^{mxn}$, entonces "$\circ$" se define como el producto $(A \circ B) \in \Re^{mxn}$, donde $(A \cdot B)_{ij} = A_{ij} \cdot B_{ij}$

[3]La partícula aprende de su experiencia.

[4]La partícula aprende de la experiencia de otras partículas.

Con el tiempo, las partículas deben converger a un peso promedio definido en función de sus atractores, es decir, p_i y g_{best}. El Algoritmo 4.1 muestra la secuencia principal de este paradigma.

Algoritmo 4.1: Algoritmo PSO

```
1  foreach partícula i do
2      Inicializar aleatoriamente v_i, x_i = p_i;
3      Actualizar g_best;
4  end
5  while no se cumpla la condición de parada do
6      foreach partícula i do
7          Actualizar la posición de i con las expresiones
8          x_i^(t+1) = x_i^(t) + v_i^(t+1) y v_i^(t+1) = v_i^(t) + c1 n1 ∘ (p_i − x_i^(t)) + c2 n2 ∘ (g_best − x_i^(t));
9          Evaluar x_i;
10         Actualizar p_i;
11         Actualizar g_best;
12     end
13 end
```

Un inconveniente que puede sufrir el PSO clásico es que las partículas pueden explotar, es decir, sus posiciones toman valores que las hacen salirse del espacio de búsqueda (por ejemplo, cuando están suficientemente lejos de p_i y g_{best}). Para solucionarlo, se puede restringir las velocidades a nivel de dimensión en un cierto valor V_{max} [97], condicionado al espacio de búsqueda. Específicamente la velocidad de las partículas en cada dimensión es limitada a esta velocidad máxima V_{max}, la cual determina cuan grande es el paso que dichas partículas pueden tomar hacia el espacio de solución. Si el valor de V_{max} es muy pequeño, las partículas no podrán explorar más allá de las regiones localmente buenas y podrían quedar atrapadas en óptimos locales. Por otro lado, si el valor de V_{max} es muy grande, las partículas podrían saltarse las mejores soluciones.

4.2.2. Peso inercial

Un avance importante para asegurar la convergencia de PSO ha sido incluir un peso (denominado inercial) a la expresión de la velocidad. Considere por ejemplo, la siguiente expresión que modifica a la anterior (4-1), incluyendo ahora el peso w:

$$v_i^{(t+1)} = wv_i^{(t)} + c_1 n_1 \circ \left(p_i - x_i^{(t)}\right) + c_2 n_2 \circ \left(g_{best} - x_i^{(t)}\right). \tag{4-3}$$

Una idea sugerente es utilizar w<1.0, con esto se asegura un decrecimiento de la velocidad con el tiempo, comportamiento que no se logra, por el contrario, si w>1.0.
De hecho, se ha podido comprobar que con este peso inercial, PSO converge en determinados casos sin el uso de V_{max} [98]. Los autores Eberhart y Shi proponen en [99] el uso de este peso, de manera que disminuya con el tiempo desde 0.9 hasta 0.4. Sin embargo, aunque este coeficiente permite en muchos casos que PSO no explote, es recomendable utilizar V_{max} [98], para un comportamiento más efectivo del algoritmo.

4.3. Método RankPSO

4.3.1. Descripción del algoritmo

En el presente apartado se describe de manera formal un nuevo método de L2R para la RI. Este método, al que denominamos RankPSO [100], está basado en PSO y es capaz de optimizar cualquier medida de evaluación usada en RI.
Para una mejor comprensión del procedimiento general de RankPSO, es importante describir cómo se realizó la representación y el ajuste de los elementos del modelo de PSO en la concepción del método propuesto.
Como nuestro algoritmo usa una función de *ranking* lineal basada en rasgos (extraídos a partir de un documento sobre la base de una consulta), el tamaño o dimensión de una partícula queda determinado por el número de rasgos. Por ejemplo, cuando se trate de los *datasets* de la colección .Gov de LETOR 3.0, el tamaño de cada partícula estará representado por un vector de 64 dimensiones, debido a que esta es la cantidad de rasgos que definen un documento en estos conjuntos de datos. Cada partícula i constituye una posible solución, la cual se representa por un vector de pesos en la función de *ranking*

$f(q_i, d_{ij}) = {x_i}^T \phi(q_i, d_{ij})$. Como función de aptitud se utilizó $R(f) = \sum_{i=1}^{m} (1 - E(\pi_i, y_i))$, la cual está definida directamente por una medida de evaluación de RI. El vector de posición registrado por la memoria global g_{best} en la última iteración, será usado como vector de pesos en la función de *ranking* que finalmente devolverá el algoritmo. En la Sección 4.4.1 puede ser consultada con mayor detalle la configuración de los parámetros para RankPSO.

Además de estas adaptaciones del modelo de PSO al problema del L2R, RankPSO incorpora un mecanismo para evitar la convergencia temprana de las partículas hacia los mínimos locales o zonas de estancamiento para la búsqueda. Este mecanismo, mide para un espacio de iteraciones la amplitud de la variación del rendimiento que van teniendo las partículas con respecto al conjunto de entrenamiento. Si este valor de amplitud está por debajo de un umbral de cambio predefinido, entonces el algoritmo aplica una estrategia para mover (sacar) las partículas de esta zona de no mejoras.

El procedimiento general de RankPSO se muestra en el Algoritmo 4.2, donde tenemos como parámetros de entrada para RankPSO los siguientes: un conjunto de entrenamiento S, una medida de evaluación E, un número de iteraciones T, un punto de referencia inicial P_1, un valor P_2 que representa un rango de iteraciones a partir de la iteración definida como P_1, y como último parámetro, un umbral de cambio th.

En un primer momento, RankPSO crea un enjambre de partículas e inicializa cada uno de los elementos de estas partículas de forma aleatoria y uniforme. Cada uno de los elementos de estos vectores de posición quedarán delimitados por el espacio predefinido para cada dimensión, el cual representa la amplitud de valores que puede tomar el peso para un determinado rasgo de la colección. Durante esta fase inicial se actualiza la memoria local p_i de cada partícula i, y la memoria global del enjambre g_{best}.

En la etapa de evolución del enjambre (paso $n^o 6$), las partículas, durante T iteraciones, se propagan en σ dimensiones por el espacio de búsqueda, con el objetivo de encontrar el mejor vector de posición g_{best}. Es decir, aquel vector de pesos más idóneo para la tarea del *ranking* en el conjunto de entrenamiento S. El valor de σ está determinado por el número de rasgos que presentan los documentos en S.

Algoritmo 4.2: Algoritmo RankPSO

Input: $S = \{(q_i, d_i, y_i)\}_{i=1}^{m}$, E, T, P_1, P_2 y th
foreach *partícula* i **do**
    Inicializar aleatoriamente v_i, $x_i = p_i$;
    Actualizar g_{best};
end
for t = *1,...,*T **do**
    foreach *partícula* i **do**
        Actualizar i con las expresiones (4-3) y (4-2);
        Evaluar x_i en S, con $f(q_i, d_{ij}) = {x_i}^T \phi(q_i, d_{ij})$ y $R(f) = \sum_{i=1}^{m} (1 - E(\pi_i, y_i))$;
        Actualizar p_i;
        Actualizar g_{best};
    end
    if P_1 = t **then**
        Salvar el valor de fitness de g_{best} en $F_{temporal}$;
        Salvar t en $T_{temporal}$;
    end
    if $T_{temporal}$ + P_2 = t **then**
        if |*(fitness* g_{best} - $F_{temporal}$*)/(*P_2*-1)*| <th **then**
            Aplicar estrategia de diversificación;
        end
        Salvar el valor de fitness de g_{best} en $F_{temporal}$;
        Salvar t en $T_{temporal}$;
    end
end
Construir la función de ranking f con el vector de posición g_{best};
Result: f

En cada iteración, cada partícula i actualiza su velocidad y a partir de este valor y su posición actual se calcula su nueva posición usando la ecuación (4-2). Para el cálculo de

la velocidad de las partículas y con la intención de mejorar la convergencia de RankPSO, utilizaremos la expresión (4-3), en la cual se adiciona el coeficiente (peso) inercial w a la ecuación clásica de la velocidad (expresión 4-1). Una selección conveniente de este peso inercial proporciona un equilibrio entre exploración global y local, donde los resultados promedios en pocas iteraciones son suficientes para encontrar una solución satisfactoria.

Para el paso n^o9 de RankPSO, la nueva posición de cada partícula es evaluada en relación al conjunto de entrenamiento S. La ecuación $R(f)$ representa la función de aptitud que usa el algoritmo, la cual está definida según una medida de evaluación E de RI y una predicción de *ranking* de nuestro método π_i. Esta predicción es obtenida a partir de la aplicación de la expresión (2-1),$f(q_i, d_{ij}) = {x_i}^T \phi(q_i, d_{ij})$, sobre el conjunto de entrenamiento S.

Una vez completado el movimiento de una partícula, se actualizan nuevamente las memorias local p_i y global g_{best}.

Para ilustrar el movimiento de una de estas partícula, y comprender el funcionamiento de RankPSO hasta este momento, analicemos el siguiente ejemplo. Supongamos que tenemos el conjunto de entrenamiento que se muestra en la Tabla 4-1, donde para la consulta de ID.1 se han recuperado cuatro documentos. Cada uno de estos documentos está representado por dos rasgos asociados a la consulta y una etiqueta de relevancia asignada por un especialista.

Etiqueta de Relevancia	**ID. Consulta**	**Rasgo n^o1**	**Rasgo n^o2**	**ID. Doc.**
1	1	0,712	0,708	D01
0	1	0,633	0,801	D02
2	1	0,750	0,923	D03
0	1	0,490	0,910	D04

Tabla 4-1: Ejemplo sencillo de conjunto de entrenamiento para el L2R.

Aplicamos entonces RankPSO sobre este conjunto de entrenamiento para construir la respectiva función de *ranking*. En este caso, utilizamos 5 partículas, que son inicializadas en un espacio bidimensional (debido a que tenemos dos rasgos extraídos por cada documento) limitado en ambas dimensiones por un rango [0,10], y la función de pérdida $R(f)$ estará basada en la medida de evaluación MAP. Luego de la primera iteración tenemos

que las partículas han quedado distribuidas según se muestra en la Figura 4-2. Analizando la información gráfica tenemos que durante la primera evaluación se percibe como la posición de la partícula x_5 resultó ser la mejor solución de todas. Esta partícula fue inicializada en la coordenada (9,8) y para ser evaluada se utilizó su vector de posición como vector de pesos en la función de *ranking* de la expresión (2-1). A partir de esta

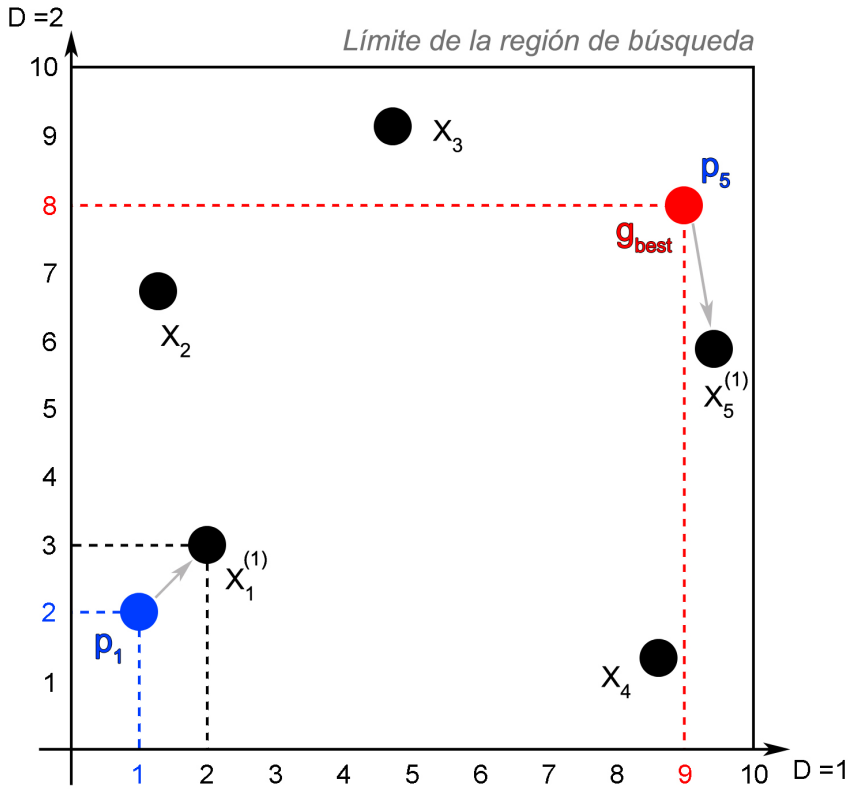


Figura 4-2: Posición de las partículas al finalizar la primera iteración.

partícula, tenemos una función de *ranking* que aplicada al documento D01 tiene la forma $f(q_1, D01) = {x_1}^T \phi(q_1, D01) = 9 * 0,712 + 8 * 0{,}708 = 12,072$. El valor resultante de esta función representa la puntuación de relevancia que asigna nuestro modelo a este documento con respecto a la consulta correspondiente. De esta misma forma calculamos la puntuación de cada documento, los ordenamos descendentemente según estos valores y calculamos MAP a partir de cómo quedaron ubicadas sus respectivas etiquetas *ground truth*. A continuación se muestra un simple esbozo de este procedimiento:

$$f\begin{pmatrix} D01 \\ D02 \\ D03 \\ D04 \end{pmatrix} = \begin{pmatrix} 12,072 \\ 12,105 \\ 14,134 \\ 11,690 \end{pmatrix} \begin{matrix} ranking \\ descendente \end{matrix} = \begin{pmatrix} D03 \\ D02 \\ D01 \\ D04 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 1 \\ 0 \end{pmatrix} MAP \begin{pmatrix} 2 \\ 0 \\ 1 \\ 0 \end{pmatrix} = 0,83.$$

Finalmente, para esta consulta tendremos un valor de precisión resultante de 0,83, que se corresponderá con un valor de pérdida $R(f) = 1 - MAP = 0,17$. Por tanto, la memoria local de x_5 tendra un valor p_5 igual a 0,17. Siguiendo este mismo proceder todas las partículas son evaluadas.

Ahora, para detallar específicamente el movimiento de una partícula, nos enfocaremos en el individuo x_1. Esta partícula fue inicializada en la posición (1,2) y en la primera iteración se movió a la posición (2,3) con una velocidad calculada igual a 1. Supongamos entonces, que los parámetros de RankPSO fueron definidos de la siguiente manera: el peso inercial w=0,9, los vectores bidimensionales n_1=0,5 y n_2=0,1; las constantes de aceleración c_1 y c_2 con un valor de 2,05.

Pasemos entonces a la segunda iteración, donde aplicaremos las expresiones (4-3) y (4-2) para calcular la velocidad y la posición que tendrá esta partícula x_1.

Para la primera dimensión (D=1):

$v_{11}^{(2)} = w*v_{11}^{(1)} + c_1*n_1(p_1 - x_{11}^{(1)}) + c_2*n_2(g_{best(1)} - x_{11}^{(1)})$,

$v_{11}^{(2)} = 0,9*1 + 2,05*0,5(1 - 2) + 2,05*0,1(9 - 2) = 1,31$,

$x_{11}^{(2)} = x_{11}^{(1)} + v_{11}^{(2)} = 2 + 1,31 = 3,31$.

Y para la segunda dimensión (D=2):

$v_{12}^{(2)} = w*v_{12}^{(1)} + c_1*n_1(p_1 - x_{12}^{(1)}) + c_2*n_2(g_{best(2)} - x_{12}^{(1)})$,

$v_{12}^{(2)} = 0,9*2 + 2,05*0,5(2 - 3) + 2,05*0,1(8 - 3) = 1,8$,

$x_{12}^{(2)} = x_{12}^{(1)} + v_{12}^{(2)} = 3 + 1,8 = 4,8$.

En estas expresiones, el valor que aparece entre paréntesis como superíndice indica el número de la iteración, el primer subíndice sugiere el número de la partícula y el segundo la dimensión. Con lo cual $v_{11}^{(2)}$ representa la velocidad que posee la partícula $n^o.1$ (x_1) en la primera dimensión para la segunda iteración.

Finalmente, como muestra la Figura 4-3, la partícula x_1 se trasladó hacia la posición (3.31,4.8). Como en esta nueva posición el valor de $R(f)$ es 0,25, igual a su memoria local

p_1 esta es actualizada, sin embargo, la memoria global no se actualiza pues el vector de posición que registra (la primera posición de x_5) tiene una pérdida menor, cuyo valor es 0,17.

De igual manera se procede a calcular la posición de cada partícula para cada nueva iteración, y como acabamos de analizar, cada vez que una determinada partícula actualiza su posición, son también actualizadas (si es el caso) su memoria local y la memoria global del enjambre.

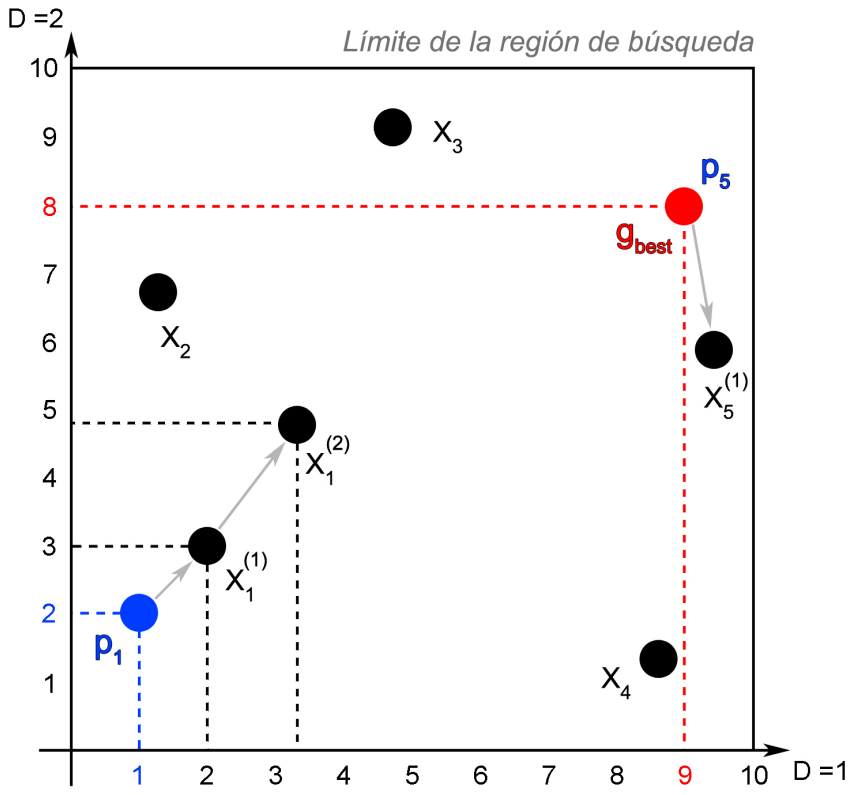


Figura 4-3: Posición de la partícula x_1 al finalizar la segunda iteración.

Siguiendo esta tónica y durante toda la evolución del enjambre, nuestro algoritmo incorpora una estrategia (ver pasos del n^o13 al 23) para evitar una convergencia temprana hacia mínimos locales o áreas de estancamiento para la búsqueda. Esta estrategia consiste en permitir al enjambre evolucionar hasta un cierto número de iteraciones, de tal modo que las partículas puedan encontrar buenas soluciones en el espacio de búsqueda. A partir de este número de iteraciones, el cual llamaremos punto de referencia inicial P_1 (por ejemplo,

iteración $n^o 20$), y durante un rango consecutivo de iteraciones P_2 (por ejemplo, 10 iteraciones) mediremos la variabilidad o mejora, τ, que van teniendo las particulas en cuanto a rendimiento sobre el conjunto de entrenamiento S.

Este valor calculado, τ, se usará como indicador para concluir si las partículas han quedado estancadas o no. Con el objetivo de controlar estos dos estados (atrapados o no atrapados), definimos un umbral de cambio, que referiremos como *th*, cuyos valores están en el rango [0,1]. Si el valor de τ es menor que *th*, podemos concluir que la búsqueda está estancada y entonces se le aplica una estrategia de diversificación a las partículas. Esta estrategia de diversificación consiste en mover las partícula hacia nuevas áreas de exploración de una forma controlada. Para lograr esta tarea, consideramos el efecto de cada valor de rasgo en el modelo. Este procedimiento persigue la idea intuitiva que el mínimo global en una determinada dimensión está frecuentemente próximo al mínimo local.

Este proceso de medir la variabilidad del rendimiento de las partículas durante P_2 iteraciones será repetido hasta que la última ronda de iteraciones del algoritmo sea completada.

Según nuestras pruebas empíricas, el parámetro P_1 puede ser fijado a un valor entre un 50 y 60 % del número de iteraciones. Para el caso de P_2, es conveniente establecer un valor entre el 15 y 20 por ciento del número de iteraciones. El valor de *th* puede ser fijado según el comportamiento del enjambre sobre una colección específica, nosotros recomendamos para los datasets que utilizaremos, valores entre 0,01 y 0,05.

Finalmente, la función de *ranking* f es construida con el vector de posición (g_{best}) obtenido en la última iteración. De esta manera, el algoritmo de aprendizaje es capaz de construir un modelo de *ranking* que optimice directamente una de las medidas de evaluación usadas en RI.

Ventajas del algoritmo: Podemos concluir que RankPSO es un método de L2R que puede ser clasificado dentro de los métodos que optimizan directamente cualquier medida de evaluación y que pertenece a la categoría de aproximación o enfoque por listas (*listwise approach*). Más específicamente, este puede ser ubicado en la subcategoría de aquellos métodos que usan tecnologías especialmente diseñadas para optimizar medidas no suaves de RI.

Además de las similitudes generales que este grupo de métodos presentan en relación a las ventajas que brinda la optimización directa de medidas de evaluación, RankPSO posee ciertas diferencias beneficiosas sustentadas en su propia concepción. RankPSO no sólo toma ventaja de las potencialidades innatas de la inteligencia de enjambre, sino que además usa un diseño simple a través de una función de *ranking* lineal e implementa una estrategia de diversificación con el objetivo de prevenir convergencias prematuras a mínimos locales.

4.3.2. Reducción de dimensionalidad usando PCA y análisis de Clúster

En la RI en general, y específicamente en el ámbito del L2R, la representación vectorial de los documentos hace posible su caracterización en términos de un conjunto de rasgos. Sin embargo, la definición y concepción de un modelo de *ranking* que sea capaz de discriminar entre documentos relevantes e irrelevantes usando esta representación constituye aún una tarea compleja.

Por otro lado, los conjuntos típicos de datos de la colección .Gov de LETOR 3.0 (una de las más usadas y referenciadas en el ámbito del L2R) para una determinada consulta, contienen una pequeña proporción de documentos relevantes comparada con aquellos irrelevantes (más del 98,5 % como promedio). Este tipo de estructura en la mayoría de los casos hace difícil la identificación de características distintivas entre aquellos documentos más marginalmente relevantes y los irrelevantes, lo cual limita la creación automática de un modelo idóneo para el L2R. Esto conlleva, además, a una ineficiente generalización del modelo de aprendizaje, lo cual atenta contra un buen rendimiento del método en posteriores etapas de prueba.

Por tanto, con el objetivo de obtener una representación más compacta de los conjuntos de datos, y a su vez garantizar la preservación de toda la información esencial, en este apartado proponemos un procedimiento de dos pasos para la reducción de los datos tomando en consideración sólo aquellos documentos irrelevantes.

Ante esta situación nos pareció prudente diseñar un procedimiento encaminado a dos objetivos fundamentales:

1. **Acentuar las diferencias entre documentos relevantes y no relevantes, a partir de alguna transformación o combinación de sus rasgos**. Para dar solución a este objetivo, se hace necesario buscar una técnica capaz de discriminar entre tales clases (etiquetas) de relevancia, que logre representar a través de rasgos y de la forma más óptima posible relaciones de proximidad y diferencias entre los documentos pertenecientes a cada una de estas etiquetas, reduciendo además el número de rasgos de tales documentos. Para facilitar este proceso, sería oportuno detectar las dependencias lineales entre los rasgos de tales documentos y sustituir los grupos de rasgos correlacionados por nuevos rasgos sin correlación. En este sentido, se hace evidente el uso de un método de extracción de rasgos que se ajuste bien a este proceder. En este caso, nos pareció oportuno utilizar el Análisis de Componentes Principales[5] (PCA, por sus siglas en inglés) [101] [102]. PCA es una técnica estadística de análisis y reducción dimensional de conjuntos de datos, que permite encontrar las proyecciones vectoriales que mejor representan la distribución de una colección de datos. Técnicamente, PCA intenta caracterizar la estructura lineal óptima de un conjunto de datos y construye una transformación lineal que escoge un nuevo sistema de coordenadas para el conjunto original de datos en el cual la varianza de mayor tamaño del conjunto de datos es capturada en el primer eje (llamado el Primer Componente Principal), la segunda varianza más grande es el segundo eje, y así sucesivamente [103]. En esencia, PCA, a partir de los vectores de rasgos del conjunto de datos, intenta construir (a través de proyecciones basadas en la varianza) un nuevo sistema de rasgos, denominado por el algoritmo Componentes Principales (PCs), que mejor represente la distribución de los documentos en función de sus etiquetas de relevancia. Mientras este conjunto completo de PCs explica toda la variabilidad en el conjunto de datos originales, un pequeño número de PCs son usualmente suficientes para explicar una gran proporción (75-90 %) de la varianza total. Además, en PCA se asume usualmente que las direcciones correspondientes

[5]Traducción literal del término inglés "Principal Component Analysis (PCA)".

de los valores propios más pequeños contienen ruido [104].

2. **Reducir la cantidad de documentos irrelevantes**. Por otro lado, para cumplir con el segundo objetivo debemos buscar la manera de determinar cuáles son aquellos documentos irrelevantes más representativos a nivel de consulta. Lo cual nos motivó a utilizar un análisis de clúster [105] usando el algoritmo de agrupamiento K-means. Para la selección del número de clústeres, es decir, el número representativo de documentos irrelevantes, usamos una estrategia basada en la aplicación del índice de Silhouette [106] (índice de validación interno), el cual mide el ajuste entre la partición impuesta por el algoritmo de agrupamiento y los datos. El índice de Silhouette puede tomar valores en el rango [-1,1]. Cuando el valor de este índice se aproxima a 1, esto significa que en la partición analizada la distribución de documentos en cada clúster es correcta en términos de la separación y compactación de los clústeres. Ocurre totalmente lo contrario cuando el valor del índice se aproxima a 0. Para el rango dado de particiones, seleccionamos aquella cuyo valor del índice de Silhouette fuese máximo.

Finalmente, a partir de estas consideraciones, diseñamos e implementamos un procedimiento que consta de los siguientes dos pasos o etapas:

1. Aplicar PCA al conjunto de datos, llevando a cabo una reducción de dimensionalidad basada en aquellos nuevos rasgos (PCs) que expliquen más del 75 % de la varianza de los datos.

2. Aplicar el algoritmo de agrupamiento K-means sobre el conjunto de documentos irrelevantes de cada consulta, seleccionando como partición (número de clúster) aquella que posea el mayor valor según el índice de Silhouette. Como representates de los clústeres, tomamos aquellos documentos irrelevantes cuyas proyecciones constituyen los vecinos más cercanos de los centroides.

Para ilustrar este proceso con un ejemplo, tomemos la primera consulta (n^o152) del conjunto de entrenamiento, que pertenece al Fold1 del conjunto de datos HP2003. Esta consulta tiene un sólo documento relevante y 999 documentos irrelevantes. Siguiendo el paso n^o1

del procedimiento aplicamos el método PCA. A partir de los resultados, se comprobó que considerando los 5 primeros PCs (es decir, nuevos rasgos) se garantizaba explicar más del 75 % de la varianza en estos datos. Por tanto, se llevó a cabo la reducción de dimensionalidad de esta consulta a este nuevo espacio de 5 PCs. Posteriormente, para el paso $n^o 2$, aplicamos el algoritmo de agrupamiento k-means sólo para los 999 documentos irrelevantes, considerando una inicialización aleatoria de los centroides y 10 réplicas del método. Este algoritmo es aplicado para un amplio rango de particiones, y con el objetivo

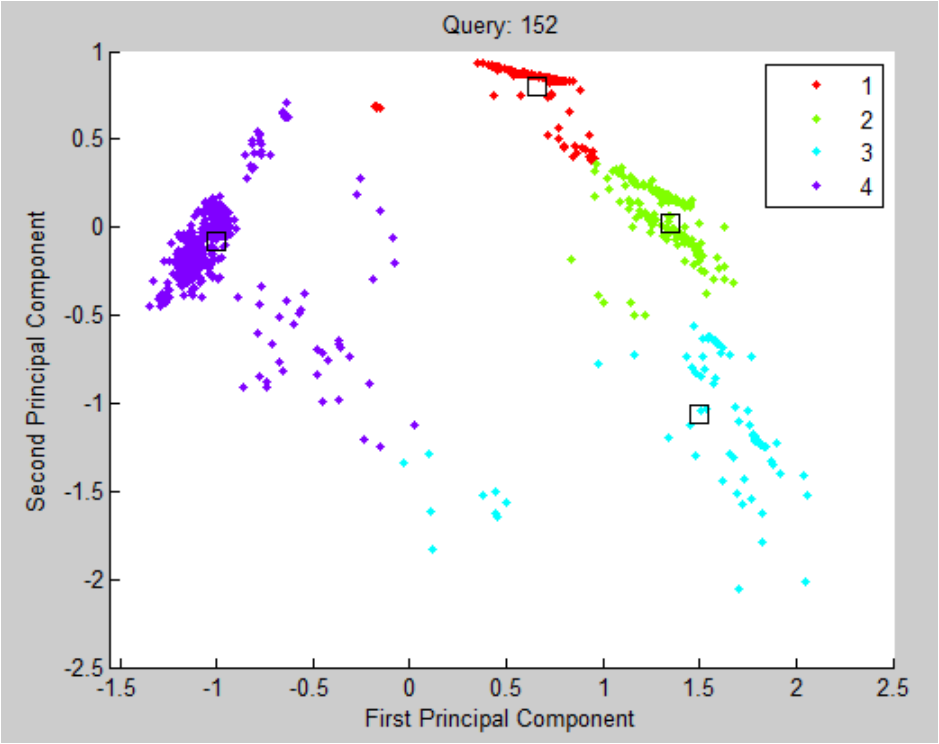


Figura 4-4: Clústeres clasificados sobre el espacio PCA para los dos primeros PCs.

de determinar el mejor agrupamiento para los datos, seleccionamos la partición con mayor valor del índice de validación interno (Silhouette). La Figura 4-4 muestra la mejor partición (cuatro clústeres) sobre el espacio PCA para las dos primeras PCs. Por su parte, la Figura 4-5 muestra el resultado de la aplicación del índice de Silhouette a esta partición.

En su interpretación, el área que representa el índice de Silhouette incluye cada uno de los documentos ubicados en los clústeres 1, 2 y 4. Comienza en la línea vertical de valor 0 y mantiene una forma larga, aproximadamente rectangular hasta casi alcanzar el valor 1.

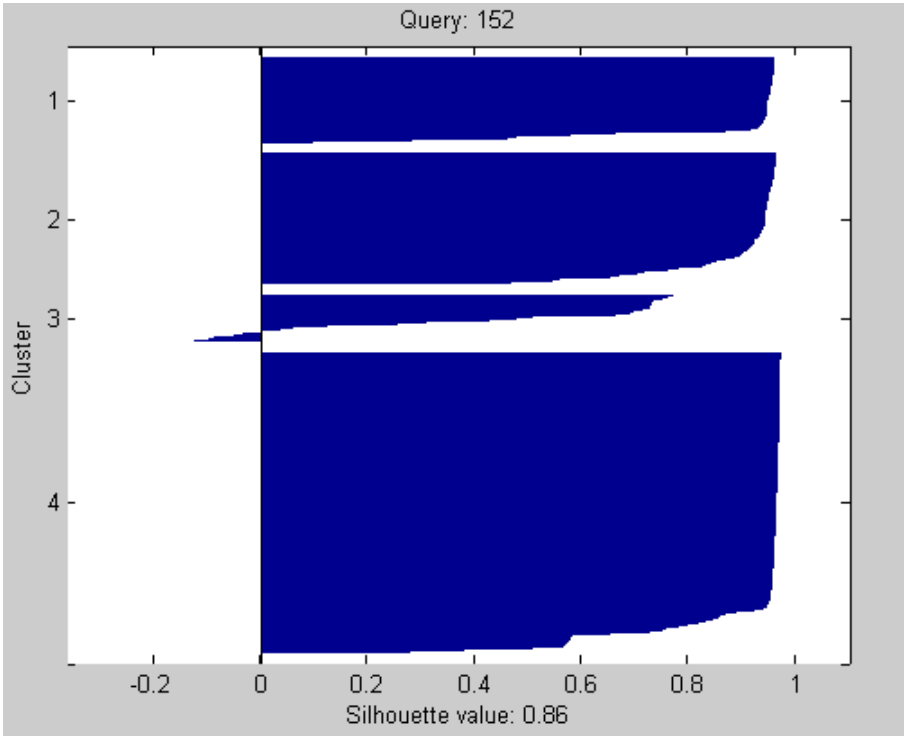


Figura 4-5: Representación de la calidad de las particiones, considerando el índice de Silhouette.

Esto indica que los clústeres son compactos y están separados unos de otros y que existe un alto grado de fiabilidad de que cada documento está ubicado en el clúster correcto. Sin embargo, el clúster 3 revela que algunos de sus documentos fueron correctamente ubicados, mientras que aquellos con un índice negativo (representados por la pequeña área que se extiende a partir de la línea vertical del punto 0 hacia la izquierda) no deben pertenecer a este grupo. Esta situación es claramente evidente en la Figura 4-4 donde varios documentos (visualizados en puntos de color cian a la izquierda) perteneciendo al clúster 3 pudiesen haber sido ubicados en el clúster 4. Finalmente, debido a esta última irregularidad, el valor del índice de Silhouette de esta partición es 0,86, el cual indica que la distribución de la mayoría de los documentos en cada clúster es correcta. Podemos decir además, que para esta partición, los clústeres son compactos y existe una considerable separación entre estos.

Finalmente, el conjunto reducido de entrenamiento consiste en el documento relevante y de cada uno de los documentos irrelevantes que representan cada clúster de la partición seleccionada.

El procedimiento descrito anteriormente es aplicado a cada consulta perteneciente a los conjuntos de entrenamiento de cada *dataset* de la colección utilizada.

La siguiente experimentación mostrará que este procedimiento tiene un impacto positivo en la tarea del L2R para los *datasets* de la colección .Gov de LETOR 3.0.

4.4. Experimentación

En esta sección, presentamos los resultados experimentales de la fase de evaluación de nuestro método propuesto en términos de su efectividad frente a los conjuntos de datos de la colección .Gov de LETOR 3.0. En este análisis seguimos los esquemas experimentales definidos en LETOR para poder realizar comparaciones directas con los principales métodos que establecen el estado del arte en L2R. Como medidas de evaluación, usamos las estándares para este campo, *Mean Average Precision* (MAP), *Normalized Discounted Cumulative Gain* (NDCG) y la *precision at n* (P@n), considerando las 10 primeras posiciones del *ranking*. Analizamos además, la robustez de nuestro algoritmo tomando en consideración la varianza entre las diferentes ejecuciones. Los rendimientos obtenidos por cada uno de los algoritmos con respecto a cada uno de los conjuntos de datos fueron analizados estadísticamente. Finalmente, se evaluó el coste computacional de RankPSO.

4.4.1. Evaluación de rendimientos obtenidos y comparación de métodos

Para poner en acción el método propuesto, establecimos los siguientes parámetros: 50 iteraciones y 40 partículas (un total de 2.000 evaluaciones). El peso inercial w fue fijado al valor 0,9, y controlado su decremento hasta 0,4. Los vectores n-dimensionales n_1 y n_2 fueron instanciados por números aleatorios en el intervalo [0,1]. Las constantes de aceleración c_1 (*particle increment*) y c_2 (*global increment*) fueron ambas instanciadas con el valor 2,05. El punto inicial de referencia P_1 fue fijado en 25, P_2 en 8 y th en 0,01. En

cuanto a la velocidad máxima, V_{max}, y siguiendo las sugerencias probadas en el trabajo [97] fue limitada al 50 % del espacio de búsqueda relativo a cada dimensión para cada partícula. Los valores de ciertos parámetros, tales como w, c_1 y c_2 fueron determinados considerando los trabajos [97,107], donde abordan el comportamiento de PSO en problemas continuos y espacios multidimensionales complejos. El ajuste final de los demás parámetros fue realizado de forma empírica siguiendo un diseño simple de experimentos, que estuvo condicionado por la sensibilidad mostrada por RankPSO en relación a la precisión alcanzada frente a las colecciones de datos de LETOR 3.0.

Definidos los parámetros, y con el objetivo de evaluar el comportamiento de nuestro método RankPSO y poder comparar los rendimientos obtenidos con las principales propuestas referenciadas en la literatura, seguimos las especificaciones experimentales publicadas en el sitio web de LETOR.

Se llevó a cabo para todas las etapas experimentales una validación cruzada de 5-*fold*. Los ficheros prefijados en los 5-*fold* de cada *dataset* fueron tomados a partir de la versión normalizada "*QueryLevelNorm*", lo cual nos permitió realizar comparaciones directas frente a los algoritmos publicados, en términos de precisión. Cada partición incluye todas las consultas del *dataset* correspondiente y consta de tres ficheros: el conjunto de entrenamiento, el conjunto de validación y el conjunto de prueba. El conjunto de entrenamiento incluye el 60 % de las consultas, el de validación un 20 % y las restantes consultas (también un 20 %) pertenecen al conjunto de prueba.

RankPSO fue comparado con todos aquellos algoritmos publicados en el sitio web de LETOR, cuyos valores de precisión están disponibles para los *datasets* de la colección .Gov. Los algoritmos seleccionados con función de *ranking* lineal fueron: Regression [108], RankSVM [59], ListNet [67], AdaRank-MAP [9], AdaRank-NDCG [9], SVMmap [21], Regression+L2reg (basado en el trabajo descrito en [109]), RankSVM-Primal [110], RankSVM-Struct [70] y SmoothRank [111]. También, fue considerado el algoritmo Frank [19], que implementa una función de *ranking* no lineal.

Todas las ejecuciones de RankPSO fueron realizadas considerando MAP como medida de evaluación en la expresión $R(f)$.

Con el objetivo de analizar el método propuesto en profundidad mientras comprobamos las potencialidades del procedimiento de reducción de dimensionalidad descrito en la Sección 4.3.2, aplicamos RankPSO en dos variantes. La primera radica en aplicar el método propuesto sin estrategia de diversificación y sin reducir la dimensionalidad de los conjuntos de entrenamiento. Los resultados obtenidos fueron etiquetados como *RankPSO*. En la segunda variante sí se aplica el procedimiento de reducción de dimensionalidad y RankPSO incorpora la estrategia de diversificación en su funcionamiento. Para este caso, usamos la etiqueta *RankPSO-Div-Red*[6].

Para esta segunda variante los conjuntos de entrenamiento fueron transformados, es decir, reducidos siguiendo el procedimiento explicado en la Sección 4.3.2. Luego, el algoritmo RankPSO ejecutó su proceso de entrenamiento sobre dichos conjuntos reducidos para cada partición. Una vez construido el modelo de *ranking* para ambas variantes fue entonces validado y probado considerando las medidas de evaluación correspondientes. Según las pautas experimentales de LETOR, los cinco resultados obtenidos (uno por cada partición) de la fase de prueba son promediados y finalmente, este valor resultante puede ser comparado directamente con los demás algoritmos publicados.

Además, para demostrar la efectividad y la influencia del procedimiento de reducción de datos en el rendimiento de un método de L2R, usamos otro algoritmo, AdaRank. Seleccionamos este algoritmo pues es utilizado como método de referencia en varias publicaciones de L2R, sus rendimientos para las colecciones de datos utilizadas son publicadas en el sitio web de LETOR, y además, contamos con los códigos fuente. Específicamente aplicamos el método AdaRank-NDCG sobre los conjuntos de entrenamiento reducidos, los resultados obtenidos fueron etiquetados como *AdaRank-NDCG-Red*.

La Tabla 4-2 muestra la precisión en el *ranking* lograda por cada uno de los métodos sobre los *datasets*: (a) TD2004, (b) TD2003, (c) NP2004, (d) NP2003, (e) HP2004 y (f) HP2003, considerando MAP como medida de evaluación. Estos resultados muestran como *RankPSO-Div-Red* alcanza los mejores rendimientos en todos los conjuntos de datos

(a) Rendimientos en TD2004

Algoritmos	**MAP**
Regression	0,2078
RankSVM	0,2237
ListNet	0,2231
AdaRank-MAP	0,2189
AdaRank-NDCG	0,1936
AdaRank-NDCG-Red	0,2025
SVMmap	0,2049
RankPSO	0,2238
RankPSO-Div-Red	**0,2399**
FRank	0,2388
Regression+L2reg	0,1992
RankSVM-Primal	0,2061
RankSVM-Struct	0,2196
SmoothRank	0,2326

(b) Rendimientos en TD2003

Algoritmos	**MAP**
Regression	0,2409
RankSVM	0,2628
ListNet	0,2753
AdaRank-MAP	0,2283
AdaRank-NDCG	0,2368
AdaRank-NDCG-Red	0,2405
SVMmap	0,2445
RankPSO	0,2492
RankPSO-Div-Red	**0,2768**
FRank	0,2031
Regression+L2reg	0,2434
RankSVM-Primal	0,2653
RankSVM-Struct	0,2713
SmoothRank	0,2695

(c) Rendimientos en NP2004

Algoritmos	**MAP**
Regression	0,5142
RankSVM	0,6588
ListNet	0,6720
AdaRank-MAP	0,6220
AdaRank-NDCG	0,6269
AdaRank-NDCG-Red	0,6321
SVMmap	0,6620
RankPSO	0,6616
RankPSO-Div-Red	**0,6914**
FRank	0,6008
Regression+L2reg	0,6866
RankSVM-Primal	0,6755
RankSVM-Struct	0,6771
SmoothRank	0,6760

(d) Rendimientos en NP2003

Algoritmos	**MAP**
Regression	0,5644
RankSVM	0,6957
ListNet	0,6895
AdaRank-MAP	0,6783
AdaRank-NDCG	0,6678
AdaRank-NDCG-Red	0,6781
SVMmap	0,6869
RankPSO	0,6812
RankPSO-Div-Red	0,6937
FRank	0,6640
Regression+L2reg	0,6824
RankSVM-Primal	0,6883
RankSVM-Struct	0,6788
SmoothRank	**0,6958**

(e) Rendimientos en HP2004

Algoritmos	**MAP**
Regression	0,5256
RankSVM	0,6675
ListNet	0,6899
AdaRank-MAP	0,7219
AdaRank-NDCG	0,6914
AdaRank-NDCG-Red	0,6992
SVMmap	0,7176
RankPSO	0,6880
RankPSO-Div-Red	**0,7220**
FRank	0,6817
Regression+L2reg	0,6301
RankSVM-Primal	0,6712
RankSVM-Struct	0,6784
SmoothRank	0,7174

(f) Rendimientos en HP2003

Algoritmos	**MAP**
Regression	0,4968
RankSVM	0,7408
ListNet	0,7659
AdaRank-MAP	0,7710
AdaRank-NDCG	0,7480
AdaRank-NDCG-Red	0,7501
SVMmap	0,7421
RankPSO	0,7492
RankPSO-Div-Red	**0,7712**
FRank	0,7095
Regression+L2reg	0,7486
RankSVM-Primal	0,7645
RankSVM-Struct	0,7625
SmoothRank	0,7633

Tabla 4-2: Rendimientos obtenidos por los diferentes algoritmos de L2R sobre los *datasets* de la colección .Gov, considerando MAP como medida de evaluación.

(excepto en NP2003, donde se ubica en la 3^{a} posición).

La precisión alcanzada por *RankPSO-Div-Red* nos permite corroborar además que la incorporación de una estrategia de diversificación y de un procedimiento para reducir la dimensionalidad de los conjuntos de entrenamiento, contribuyen satisfactoriamente en que RankPSO mejore su rendimiento en la tarea del L2R sobre todos los *datasets* evaluados.

Por otro lado, los resultados obtenidos por *AdaRank-NDCG-Red* reafirman la eficiencia de este procedimiento de reducción de datos para propiciar mejoras en el rendimiento de otros métodos de L2R sobre los *datasets* considerados. Estos resultados representan un nuevo enfoque que puede ser usado en el área del L2R para mejorar el rendimiento de los algoritmos y como base para trabajos futuros.

Las tablas del Apéndice B muestran en detalle los valores de precisión (según P@n de la posición 1 a la 10) obtenidos en el *ranking* por cada uno de estos algoritmos publicados y RankPSO sobre TD2004, TD2003, NP2004, NP2003, HP2004 y HP2003, respectivamente. Para una mejor interpretación de los resultados expuestos, en estas tablas, incorporamos indistintamente los símbolos (+), (-) o (=) a la izquierda del valor de cada método para cada columna. Esta notación nos indica comparativamente como se comportan todos los métodos publicados con respecto a *RankPSO-Div-Red*, en esencia, si el rendimiento es mejor, es peor o es igual.

Finalmente, estos resultados muestran que *RankPSO-Div-Red* alcanza siempre el mejor o segundo mejor rendimiento en la primera posición del *ranking* en comparación con todos los métodos y conjuntos de datos evaluados. Manteniendo además, una buena estabilidad de precisión en las restantes posiciones del *ranking* para todos los conjuntos de datos, a diferencia de otros métodos de L2R que presentan comportamientos excelentes en algunos casos y muy malos para otros. En el Apéndice B se detalla la discusión de estos resultados a nivel de *dataset*.

Entretanto, la Tabla 4-3 muestra el comportamiento de los valores de precisión en el *ranking* obtenidos por cada método en (a) TD2004, (b) TD2003, (c) NP2004, (d) NP2003, (e) HP2004 y (f) HP2003, bajo los términos de las medidas NDCG@1, NDCG@5 y NDCG@10.

[6]Los resultados preliminares muestran que la combinación de estas dos estrategias aportan más al rendimiento del método que el usarlas de forma individual

(a) Rendimientos en TD2004

Algoritmos	NDCG@1	NDCG@5	NDCG@10
Regression	(-)0,36	(-)0,3257	(-)0,3031
RankSVM	(-)0,4133	(-)0,324	(-)0,3078
ListNet	(-)0,36	(-)0,3325	(-)0,3175
AdaRank-MAP	(-)0,4133	(-)0,3602	(-)0,3285
AdaRank-NDCG	(-)0,4267	(-)0,3514	(-)0,3163
AdaRank-NDCG-Red	(-)0,427	(-)0,352	(-)0,317
SVMmap	(-)0,2933	(-)0,3007	(-)0,2907
RankPSO	(-)0,36	(-)0,3502	(-)0,3227
RankPSO-Div-Red	0,5067	0,371	0,3346
FRank	(-)0,4933	(-)0,3629	(-)0,3331
Regression+L2reg	(-)0,2933	(-)0,2917	(-)0,2832
RankSVM-Primal	(-)0,3067	(-)0,3062	(-)0,2913
RankSVM-Struct	(-)0,3467	(-)0,3192	(-)0,309
SmoothRank	(-)0,4	(-)0,3555	(-)0,3343

(b) Rendimientos en TD2003

Algoritmos	NDCG@1	NDCG@5	NDCG@10
Regression	(-)0,32	(-)0,2984	(-)0,3263
RankSVM	(-)0,32	(-)0,3621	(-)0,3461
ListNet	(+)0,4	(-)0,3393	(-)0,3484
AdaRank-MAP	(-)0,26	(-)0,3029	(-)0,3069
AdaRank-NDCG	(-)0,36	(-)0,2939	(-)0,3036
AdaRank-NDCG-Red	(-)0,36	(-)0,2940	(-)0,3041
SVMmap	(-)0,32	(-)0,3318	(-)0,3282
RankPSO	(-)0,34	(-)0,3205	(-)0,3240
RankPSO-Div-Red	0,39	0,3647	0,3544
FRank	(-)0,3	(-)0,2468	(-)0,269
Regression+L2reg	(-)0,34	(-)0,3381	(-)0,3297
RankSVM-Primal	(-)0,32	(+)0,3659	(+)0,3571
RankSVM-Struct	(-)0,34	(+)0,3654	(-)0,3467
SmoothRank	(-)0,38	(-)0,3345	(-)0,3367

(c) Rendimientos en NP2004

Algoritmos	NDCG@1	NDCG@5	NDCG@10
Regression	(-)0,3733	(-)0,6135	(-)0,6536
RankSVM	(-)0,5067	(-)0,7957	(-)0,8062
ListNet	(-)0,5333	(-)0,7965	(-)0,8128
AdaRank-MAP	(-)0,48	(-)0,731	(-)0,7497
AdaRank-NDCG	(-)0,5067	(-)0,7122	(-)0,7384
AdaRank-NDCG-Red	(-)0,5067	(-)0,7125	(-)0,739
SVMmap	(-)0,52	(-)0,7869	(-)0,8079
RankPSO	(-)0,546	(-)0,7998	(-)0,8131
RankPSO-Div-Red	0,56	0,8096	0,8235
FRank	(-)0,48	(-)0,687	(-)0,7296
Regression+L2reg	(+)0,5733	(-)0,7774	(-)0,804
RankSVM-Primal	(=)0,56	(-)0,7719	(-)0,795
RankSVM-Struct	(=)0,56	(-)0,7746	(-)0,7977
SmoothRank	(-)0,5467	(-)0,7825	(-)0,8075

(d) Rendimientos en NP2003

Algoritmos	NDCG@1	NDCG@5	NDCG@10
Regression	(-)0,4467	(-)0,6423	(-)0,6659
RankSVM	(=)0,58	(-)0,7823	(-)0,8003
ListNet	(-)0,5667	(-)0,7843	(-)0,8018
AdaRank-MAP	(=)0,58	(-)0,7482	(-)0,7641
AdaRank-NDCG	(-)0,56	(-)0,7447	(-)0,7672
AdaRank-NDCG-Red	(-)0,565	(-)0,7447	(-)0,7672
SVMmap	(-)0,56	(+)0,7881	(-)0,7975
RankPSO	(=)0,58	(-)0,7520	(-)0,7816
RankPSO-Div-Red	0,58	0,785	0,8025
FRank	(-)0,54	(-)0,7595	(-)0,7763
Regression+L2reg	(-)0,5467	(+)0,7887	(=)0,8025
RankSVM-Primal	(-)0,5733	(-)0,7748	(-)0,7894
RankSVM-Struct	(-)0,5533	(-)0,7789	(-)0,7955
SmoothRank	(=)0,58	(+)0,7892	(-)0,7986

(e) Rendimientos en HP2004

Algoritmos	NDCG@1	NDCG@5	NDCG@10
Regression	(-)0,3867	(-)0,613	(-)0,6468
RankSVM	(-)0,5733	(-)0,7512	(-)0,7687
ListNet	(-)0,6	(+)0,7694	(-)0,7845
AdaRank-MAP	(-)0,6133	(+)0,8277	(+)0,8328
AdaRank-NDCG	(-)0,5867	(+)0,792	(+)0,8057
AdaRank-NDCG-Red	(-)0,589	(+)0,7931	(+)0,806
SVMmap	(-)0,63	(+)0,8011	(+)0,8062
RankPSO	(-)0,59	(+)0,7667	(-)0,784
RankPSO-Div-Red	0,667	0,7618	0,7922
FRank	(-)0,6	(-)0,7486	(-)0,7615
Regression+L2reg	(-)0,5333	(-)0,6979	(-)0,7188
RankSVM-Primal	(-)0,5733	(-)0,7528	(-)0,772
RankSVM-Struct	(-)0,5867	(-)0,7523	(-)0,7666
SmoothRank	(-)0,6133	(+)0,8169	(+)0,8221

(f) Rendimientos en HP2003

Algoritmos	NDCG@1	NDCG@5	NDCG@10
Regression	(-)0,42	(-)0,5463	(-)0,5943
RankSVM	(-)0,6933	(-)0,7954	(-)0,8077
ListNet	(-)0,72	(+)0,8298	(+)0,8372
AdaRank-MAP	(-)0,7333	(+)0,8252	(+)0,8384
AdaRank-NDCG	(-)0,7133	(-)0,8006	(-)0,806
AdaRank-NDCG-Red	(-)0,7135	(-)0,8014	(-)0,807
SVMmap	(-)0,7133	(-)0,7922	(-)0,7994
RankPSO	(-)0,7156	(-)0,8016	(-)0,807
RankPSO-Div-Red	0,753	0,8142	0,8261
FRank	(-)0,6533	(-)0,778	(-)0,797
Regression+L2reg	(-)0,6933	(=)0,8142	(-)0,8216
RankSVM-Primal	(-)0,74	(-)0,8081	(-)0,818
RankSVM-Struct	(-)0,74	(-)0,8074	(-)0,8162
SmoothRank	(-)0,7133	(+)0,8287	(+)0,8325

Tabla 4-3: Rendimiento de algoritmos estudiados, considerando NDCG@1, NDCG@5 y NDCG@10.

En este caso, el rendimiento de nuestro método propuesto reafirma su condición de mejor o segundo mejor método en la primera posición del *ranking*, considerando ahora los valores de precisión obtenidos según la medida de evaluación NDCG. También, si consideramos el comportamiento de *RankPSO-Div-Red* en las posiciones del *ranking* de la 5 a la 10 para todos los *datasets*, es notable percibir como nuestro método se mantiene siempre en las tres primeras posiciones en comparación con los demás algoritmos.

Como ya hemos concluido, *RankPSO-Div-Red* presenta resultados muy satisfactorios y

prometedores frente a los datasets evaluados y en comparación con los demás métodos de L2R. Sin embargo, a pesar de este mérito, tales resultados publicados constituyen valores promedios de precisión. Por tanto, sería interesante, analizar el grado de variabilidad de todos los rendimientos de *RankPSO-Div-Red*, con la intención de corroborar a un nivel más profundo el grado de estabilidad que alcanza.

Para ello, decidimos calcular el coeficiente de variación (CV) que presentan todos los rendimientos obtenidos por *RankPSO-Div-Red* bajo diferentes situaciones, es dedir, medidas de evaluación y *datasets*. La media aritmética fue calculada a partir del resultado de 30 ejecuciones llevadas a cabo por nuestro algoritmo en cada conjunto de datos. Luego, tomando en consideración la varianza entre las diferentes ejecuciones, se calculó el CV, definido como la desviación estándar dividida por la media aritmética y multiplicada por 100.

DATASET	MAP	P@1	P@5	P@10	NDCG@1	NDCG@5	NDCG@10
HP2003	1,12	0,67	1,26	1,00	0,67	0,46	0,32
HP2004	1,23	0,96	1,70	1,71	0,82	0,51	0,27
NP2003	1,18	0,96	1,40	1,04	0,84	0,44	0,24
NP2004	1,15	0,92	1,20	1,56	0,82	0,40	0,25
TD2003	1,28	1,05	1,62	0,97	1,15	0,98	0,56
TD2004	1,35	1,04	1,51	1,02	1,04	0,91	0,71

Tabla 4-4: Coeficiente de variación (%) alcanzado por *RankPSO-Div-Red* en todos los *datasets* y considerando MAP, P@n y NDCG@n como medidas de evaluación.

La Tabla 4-4 muestra en porcentaje los valores del CV para todos los *datasets*, y en relación a las principales medidas de evaluación referenciadas en L2R (MAP, P@1, P@5, P@10, NDCG@1, NDCG@5 y NDCG@10). El CV promedio considerando todos los conjuntos de datos y medidas de evaluación resultó ser 0.95 %. Este valor indica que los valores de precisión alcanzados por el algoritmo propuesto poseen un bajo grado de variabilidad, lo que hace que dichos rendimientos tengan un mayor grado de homogeneidad, independientemente de los conjuntos de datos evaluados y medidas usadas.

4.4.2. Análisis estadístico

En este apartado realizamos un análisis más profundo de los resultados obtenidos por cada uno de los métodos estudiados sobre los conjuntos de datos evaluados y medidas

utilizadas. Con tal objetivo, aplicamos una serie de pruebas estadísticas para determinar la significación de la precisión obtenida por cada uno de los métodos a nivel de consulta. Pruebas no paramétricas fueron aplicadas usando un modelo estadístico de comparación de medias para muestras relacionadas o pareadas. Este modelo fue aplicado porque se ajusta bien a la necesidad de comparar diferentes algoritmos en un mismo grupo de consultas; todo lo cual permitió llevar a cabo un estudio comparativo de los rendimientos de cada algoritmo con un alto nivel de veracidad y exactitud.

Específicamente, se realizó un análisis de varianza de dos vías (*two-way ANOVA*) [112,113], donde se aplicó en un primer momento la prueba de Friedman (*Friedman test*) [114, 115]. Si el resultado de la prueba es significante, entonces podemos afirmar con un alto grado de certeza que existen diferencias significativas entre las medias de, al menos, dos de las muestras relacionadas. En tal caso, para determinar entonces qué algoritmos tienen diferencias significantivas entre sí y además quién supera a quién, es necesario analizar si la distribución de los datos concurre de tal manera que existe, o no existe, predominio de aumentos o disminuciones entre sus diferencias. Con la intención de verificar esta hipótesis, la prueba de los rangos con signo de Wilcoxon [116] (*Wilcoxon signed-rank test*) nos permite comparar la media de dos muestras relacionadas y determinar si existen diferencias entre ellas.

Para llevar a cabo este análisis usamos un intervalo de confianza del 95 % cuando aplicamos la prueba de Friedman y para la prueba de Wilcoxon el intervalo de confianza fue fijado a un 99 %. Pero además, con la intención de resaltar pequeñas diferencias o tendencias estadísticas que se aproximan a ser significantes entre los algoritmos y así poder tener una idea un poco más tentativa y particular del comportamiento de los algoritmos, en la interpretación de los resultados arrojados por la prueba de Wilcoxon consideramos lo siguiente a partir del trabajo [117]:

- Altamente significativo, una significación menor que 0,01.
- Significativo, un resultado de significación menor que 0,05 y mayor que 0,01.
- Medianamente significativo, un resultado menor que 0,1 y mayor que 0,05.
- No significativo, un resultado mayor que 0,1.

Estas pautas tienen cierto nivel de subjetividad, pero se sustentan en la idea de no sólo saber tajantemente si existen diferencias entre los algoritmos, sino además, de llegar a conocer con un cierto grado de detalle cuan afirmadas y delimitadas puedan estar.

Comenzando el análisis estadístico, tenemos que, como variable dependiente, usaremos la precisión a nivel de consulta alcanzada por cada algoritmo frente a cada conjunto de datos (por ejemplo, para HP2003 tendremos 150 valores de precisión por cada algoritmo). Para el cálculo de la precisión consideramos MAP como medida de rendimiento pues esta representa la precisión media promedio de todo el *ranking* correspondiente a cada consulta y no sólo hace alusión, como otras medidas, a las primeras posiciones.

Aplicamos entonces, la prueba de Friedman para determinar si existen diferencias significativas entre las medias de las muestras de precisión de alguno de los algoritmos de L2R, para alguno de los conjuntos de datos. En la interpretación de los resultados (ver detalles en Apéndice C), debido a que el nivel crítico (significancias asintóticas) asociado a cada uno de los casos es menor que 0.05, podemos rechazar la hipótesis nula de igualdad de medias y concluir que la eficiencia en el *ranking* sobre HP2003, NP2003, HP2004, NP2004, TD2003 y TD2004, considerando MAP como medida de rendimiento, no es la misma para ninguno de los diez algoritmos de L2R. Es decir, que existen diferencias significativas entre al menos dos algoritmos en cada uno de los seis conjuntos de datos.

Por tanto, pasamos a aplicar la prueba de Wilcoxon para determinar puntualmente entre qué algoritmos están dadas las diferencias significativas, y para qué conjuntos de datos.

En general, los resultados estadísticos obtenidos en los *datasets*: HP2003, HP2004 y NP2003, muestran como todos los métodos de L2R mejoran significativamente el rendimiento de Regression. En el caso de NP2004, todos los métodos en un mayor o menor grado mejoran significativamente el rendimiento mostrado por RankBoost y Regression. Para el conjunto de datos TD2003, es perceptible que el comportamiento del rendimiento de *RankPSO-Div-Red* y ListNet superan con un alto nivel de significación a todos los métodos de L2R. En TD2004, y en particular, las medias de precisión alcanzadas por AdaRank-NDCG son inferiores estadísticamente a todos los demás métodos de L2R (excepto Regression), y en este *dataset*, y en contraposición al rendimiento mostrado en NP2004, RankBoost es capaz de superar estadísticamente el rendimiento de todos los

demás métodos de aprendizaje.

Si analizamos como los métodos se manifiestan frente a cada *dataset*, podemos determinar que ciertos métodos (por ejemplo, Regression) en la mayoría de los casos tiene valores de precisión muy bajos. Otros métodos tales como RankBoost tienen un comportamiento inestable pues alcanzan altos valores en algunos casos (por ejemplo, TD2004) y muy bajos en otras situaciones (como en NP2004). En general, la mayoría de los métodos que optimizan directamente medidas de evaluación tuvieron resultados competitivos, pues indistintamente para los diversos conjuntos de datos, mantienen un balance de mejorías y empeoramientos de sus rendimientos en las diferentes posiciones del *ranking*.

Finalmente, podemos concluir que *RankPSO-Div-Red* alcanza el mejor valor de rango promedio en todos los *datasets* utilizados (ver tablas del Apéndice C) y manifiesta también la mejor estabilidad en términos de precisión en el *ranking* para todos los conjuntos de datos utilizados y en comparación con todos los métodos estudiados. Al final del Apéndice C, también se detallan, entre cuales algoritmos se manifiestan diferencias y si son medianamente significativas, significativas o altamente significativas.

Los métodos SVMmap, Regression+L2reg, RankSVM-Primal, RankSVM-Struct y SmoothRank no fueron considerados en el análisis estadístico debido a que sus valores de precisión a nivel de consulta para cada uno de los *datasets* evaluados no fueron publicados en el sitio web de LETOR.

4.4.3. Análisis del coste computacional

El coste computacional del método RankPSO fue evaluado considerando el tiempo total de ejecución del algoritmo, donde también se incorporó para hacer más interesante y completo el análisis los tiempos empleados por el procedimiento de reducción de dimensionalidad propuesto. En esencia, se combinaron indistintamente el tiempo de preprocesamiento de los *datasets*, el tiempo de procesamiento empleado para la construcción del modelo de *ranking* y el tiempo usado para ordenar todos los documentos en el conjunto de prueba.

En este sentido, nos enfocaremos en comparar los diferentes tiempos de ejecución para cada *dataset*, considerando el rendimiento de RankPSO frente a los conjuntos de entrenamiento

tanto originales (sin modificación) como reducidos.

En consecuencia a esta premisa, y para una mejor comprensión, cuando RankPSO es aplicado a los conjuntos de entrenamiento originales, los resultados obtenidos serán etiquetados como $RankPSO_{Original}$; y cuando se enfrente a los conjuntos de entrenamiento reducidos, usaremos la etiqueta $RankPSO_{Reducido}$.

En todas las evaluaciones, usaremos los ficheros de entrenamiento y prueba pertenecientes al Fold1 de cada conjunto de datos. Como declaramos en la Sección 4.4.1, se establecieron como parámetros del método propuesto 50 iteraciones y 40 partículas, un total de 2000 evaluaciones.

El experimento fue desarrollado sobre una arquitectura Intel(R) Core(TM) i3 2.53 GHz de procesador y 8.0 GBytes de RAM.

Inicialmente medimos el tiempo que emplea el procedimiento propuesto para la reducción de dimensionalidad, en cada uno de los dataset evaluados. La Tabla 4-5 nos muestra estos resultados de tiempo obtenidos para cada una de las dos etapas de preprocesamiento, donde por ejemplo, el valor 0,817 indica la cantidad total de segundos empleadas en aplicar PCA a las 50 consultas del conjunto de datos de entrenamiento TD2003. Por su parte, la columna encabezada con la etiqueta "TOTAL" resume el tiempo total de preprocesamiento para cada *dataset*.

DATASET	PCA	K-MEANS	TOTAL
TD2003	0,817	5.197,88	5.198,697
TD2004	1,026	6.966,97	6.967,996
HP2003	1,256	9.313,76	9.315,016
HP2004	1,210	6.921,60	6.922,810
NP2003	2,236	15.062,36	15.064,596
NP2004	0,676	4.502,87	4.503,546

Tabla 4-5: Tiempo de preprocesamiento (en segundos) empleado en cada *dataset* estudiado.

Por su parte, la Tabla 4-6 muestra el tiempo total empleado por $RankPSO_{Original}$ y por $RankPSO_{Reducido}$ (incluyendo el tiempo de preprocesamiento) para procesar los 5 *Fold* de cada *dataset*. Además, podemos apreciar en la última columna, que el porcentaje de tiempo

ahorrado por $RankPSO_{Reducido}$ es altamente significativo y considerable con respecto a la versión original.

Dataset	$RankPSO_{Original}$	$RankPSO_{Reducido}$ (+ preprocesamiento)	Reducción de tiempo (%)
TD2003	70.060	16.236	76,8
TD2004	130.685	21.829	83,3
HP2003	241.425	29.835	87,6
HP2004	124.845	21.513	82,8
NP2003	152.840	46.574	69,5
NP2004	134.630	14.461	89,3

Tabla 4-6: Tiempo total de procesamiento (en segundos) empleado en cada *dataset* para cada variante.

Finalmente, concluimos que generalmente para 2.000 evaluaciones aplicadas a los conjuntos de datos tratados, y con un conocimiento previo de los rendimientos de RankPSO, el procedimiento de reducción de dimensionalidad propuesto usando PCA y análisis de clúster con k-means demuestra ser factible y eficaz para la tarea del *ranking* debido al hecho que no sólo mejora el rendimiento del algoritmo, sino que además permite una disminución considerable del tiempo requerido para construir y el evaluar el modelo de aprendizaje.
Nosotros no podemos llevar a cabo un estudio comparativo, ni realizar ningún comentario sobre el tiempo de procesamiento de los algoritmos publicados en el sitio web de LETOR, pues esta información no está disponible, ni referenciada en este sitio.

4.5. Conclusiones

En el presente capítulo propusimos un nuevo método de L2R denominado RankPSO.
Esta propuesta basada en PSO presenta las siguientes ventajas: es fácil de implementar, permite la optimización directa de cualquier medida de evaluación utilizada en RI, considera en su concepción una estrategia para evadir la convergencia temprana en mínimos locales, el modelo de *ranking* construye una función lineal, y a diferencia de otras propuestas de optimización directa, presenta un comportamiento estable para todos los conjuntos de datos y medidas de evaluación utilizadas en el estudio experimental.

Analizando los resultados de la experimentación y comparando los rendimientos alcanzados por RankPSO en relación a los criterios de evaluación y los métodos que definen el estado del arte en L2R, podemos concluir que RankPSO presenta un rendimiento similar a los mejores métodos que aplican una optimización directa, pero con la ventaja de que siempre alcanza la primera o segunda mejor posición en el *ranking* y que en todos los casos resulta más estable en términos de precisión con respecto a todos los métodos comparados, medidas de evaluación y conjuntos de datos.

Podemos afirmar además, que para los conjuntos de datos de la colección .Gov de LETOR, RankPSO mejora significativamente el rendimiento de los principales métodos de L2R evaluados, y alcanza una mejor estabilidad en términos de precisión en el *ranking* frente a todos los casos. Además, verificamos que para todos los conjuntos de datos y medidas de evaluación, RankPSO alcanza en su rendimiento un coeficiente de variación promedio de 0.95 %; lo cual nos indica que nuestro algoritmo puede ser considerado como estable en estos casos.

Por otro lado, los rendimientos de nuestro método y de AdaRank-NDCG demostraron que la aplicación del procedimiento propuesto para la reducción de la dimensionalidad de los datos, combinando PCA y análsis de clúster con k-means, resulta factible y eficaz para la tarea del *ranking* y que no sólo contribuye a mejorar el rendimiento de los métodos de L2R sino que además le permite a los modelos de aprendizaje disminuir el tiempo computacional empleado durante la fase de entrenamiento.

Finalmente, los resultados muestran las mejoras que se pueden lograr con el uso de PSO como algoritmo bioinspirado en el diseño de un método de L2R y las ventajas que subyacen en las técnicas de reducción de dimensionalidad combinadas y adaptadas para facilitar y mejorar las tareas de *ranking*.

Capítulo 5

Procedimiento de Búsqueda del Pescador: una nueva metaheurística de optimización global

"(...) el alma de la ciencia son las ideas".
— Ruy Pérez Tamayo

5.1. Introducción

En este capítulo se presenta una nueva metaheurística denominada Procedimiento de Búsqueda del Pescador. Este algoritmo está concebido para resolver problemas de optimización global combinando la búsqueda guiada y la búsqueda local.

Se describe también una fase de parametrización donde se analiza el comportamiento de los parámetros de este algoritmo considerando las cinco funciones de prueba de De Jong [118], y se establecen recomendaciones generales para su uso.

Finalmente, se evalúan las funcionalidades de la metaheurística mediante un conjunto de funciones de referencia utilizadas para validar métodos heurísticos. En esta fase de prueba comparamos los rendimientos, la capacidad de convergencia a valores mínimos globales y el coste de tiempo, de nuestro algoritmo frente a tres métodos heurísticos referenciados en la literatura: Optimización con enjambre de partículas (PSO, *Particle Swarm Optimization*), Procedimientos de búsqueda basados en funciones voraces aleatorizadas que se adaptan (GRASP, *Greedy Randomized Adaptive Search Procedures*), y Evolución Diferencial (DE, *Differential Evolution*).

Como conclusión, podemos decir que el Procedimiento de Búsqueda del Pescador mostró ser una metaheurística simple y robusta que consigue buenas soluciones para todos los problemas teóricos evaluados.

5.2. Estrategia intuitiva de la pesca

Desde tiempos ancestrales, la pesca ha constituido una de las actividades económicas fundamentales de muchos pueblos del mundo. El oficio y el arte de pescar consiste en capturar y extraer de su medio natural (océanos, mares, lagos y ríos) peces u otras especies acuáticas (crustáceos, moluscos, etcétera).

Para comprender el comportamiento que manifiesta un pescador en su labor cotidiana, analizaremos un ejemplo que iremos ilustrando a través de la Figura 5-1.

Supongamos una situación real, en la que tenemos un pescador que decide ir a pescar a un lago de gran extensión. Este pescador cuenta con una barca y varias redes de pesca. Pero él sabe que le resultaría imposible en un período de tiempo corto, quizás un día o algunas horas, explorar toda la zona en busca de los mejores ejemplares (peces grandes). Por tanto, como todo pescador con experiencia, lo primero que hace es identificar y establecer visualmente, aquellos posibles caladeros o pesqueros más apropiados, es decir, aquellas áreas del lago donde pueda disponer sus redes de pesca y que al parecer pudiese tener una buena captura.

Siguiendo esta idea en la Figura 5-1, tenemos cinco puntos de captura: x_1, x_2 ,..., x_5. El pescador se sitúa con su barca en el punto x_1 y lanza su malla, los peces capturados chapotean cuando se enmallan y le indican al pescador que resulta prometedor moverse para el punto x_1' donde vuelve a lanzar su malla. Una vez realizados todos los lanzamientos que entienda conveniente en este punto de captura, se traslada hacia el siguiente punto de captura x_2 y reactiva su mecanismo de pesca. Siguiendo este procedimiento simple, el pescador recorre todos los puntos de captura, siempre grabando en su memoria cómo se comportó la pesca en cada uno de los caladeros para cuando vuelva a dichos puntos poder situarse en mejores posiciones. En toda esta actividad pesquera, el buen pescador utiliza, según el propósito que persiga o como se vaya comportando la pesca, diversas mallas de distintos tamaños.

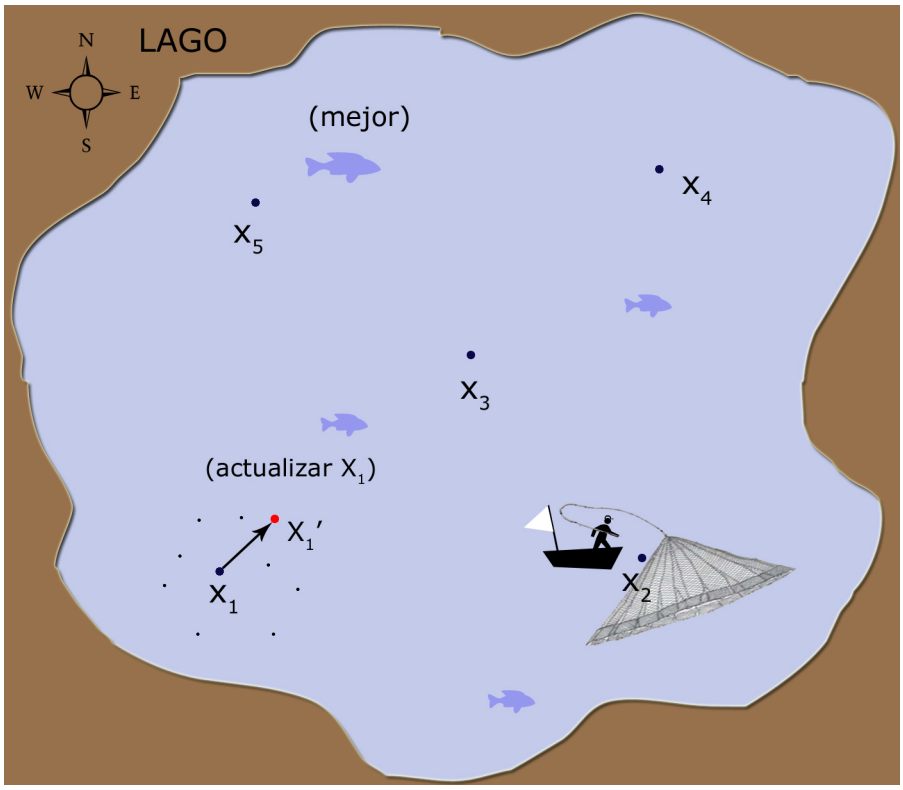


Figura 5-1: Esquema gráfico que representa el movimiento del pescador en un espacio unidimensional.

5.3. Descripción del algoritmo

El Procedimiento de Búsqueda del Pescador (FSP, *Fisherman Search Procedure*) [119], es un método de optimización global, inspirado en el comportamiento cognitivo[1] y las destrezas observadas en un pescador tradicional (como el del ejemplo analizado). Esta nueva metaheurística fue diseñada con el propósito de desarrollar soluciones útiles y prácticas para una variedad de problemas de optimización. Con este objetivo FSP explora nuevas soluciones combinando la búsqueda guiada y la búsqueda local.

Primeramente para comprender como fue concebido FSP, estableceremos el paralelo entre algunas partes de la actividad de la pesca y su aplicación a un modelo de optimización.

Apoyándonos en el ejemplo de la sección anterior, se puede definir como primera analogía

[1]Con este término nos referimos al conocimiento o cúmulo de información que se dispone gracias a un proceso de aprendizaje o a la experiencia.

que la extensión del lago donde pesca este pescador representará el espacio de búsqueda en un problema de optimización. Los caladeros o puntos de captura serán modelados como vectores de posición dentro del espacio de búsqueda. Estos vectores de posición van a constituir el conjunto inicial de posibles soluciones para el problema en cuestión. Por otro lado, la acción de lanzar la malla o red de pesca desde un punto de captura se va a corresponder, en el nuevo modelo, a evaluar en la vecindad el *fitness* de varios vectores de posición con la finalidad de encontrar una mejor solución que la hallada hasta el momento por el vector de referencia (punto desde donde se lanza la red de pesca). La amplitud que abarcará esta vecindad figurará como el tamaño de la malla, el cual estará condicionado a un valor real que denominaremos coeficiente de amplitud en el modelo de optimización. A su vez en este modelo la malla será referenciada con un valor entero, que indicará la cantidad de vectores de posición que serán evaluados en la vecindad de un punto de captura. Al igual que hace el pescador tradicional durante su desempeño en esta labor, el modelo de optimización registrará la mejor solución (vector de posición) encontrado hasta el momento en cada punto de captura y a nivel de toda la zona de pesca (espacio de búsqueda). El número de iteraciones del método representará la cantidad de rondas de pesca que el pescador lleva a cabo incluyendo todos sus puntos de pesca. En la Tabla 5-1 se muestra un resumen de todas estas equivalencias entre ambos modelos: físico y de optimización.

Modelo real		**Modelo de optimización**
Extensión del Lago	⇒	Espacio de búsqueda
Punto de captura	⇒	Vector de posición que constituye una solución al problema
Malla o red de pesca	⇒	Número de vectores de posición relativos por vecindad a un punto de captura
Tamaño de la malla	⇒	Definido por un coeficiente de amplitud
Lanzamientos de la malla	⇒	Número de veces que se genera y evalúa una malla en la vecindad de un punto de captura
Memorias del pescador	⇒	Mejor solución a nivel de punto de captura y global
Rondas de pesca	⇒	Número de iteraciones a ejecutar

Tabla 5-1: Equivalencias del pescador entre el modelo físico y el modelo de optimización.

Cuando extrapolamos las formas y estrategias seguidas por este pescador para acometer la pesca, se debe considerar que en el mundo real este pescador busca los mejores peces y tiene un tiempo para ello. En términos de optimización se trata de buscar una solución al problema lo suficientemente buena en el menor tiempo computacional. Además, en el modelo de optimización, la función objetivo será la que nos indique cuán buena va siendo la pesca y qué decisiones tomar en un momento determinado.

Una vez sentadas las premisas conceptuales de la idea y el modelo, pasemos a presentar formalmente la metaheurística FSP.

Este algoritmo en sus primeros pasos define un conjunto de N puntos de captura en toda la región de pesca (espacio de búsqueda para el problema en cuestión). Básicamente cada punto de captura está compuesto por un vector de posición, x_i, y una memoria de la mejor solución encontrada por el pescador en la vecindad del punto de captura, p_i. Se tiene que $x_i \in X$, donde $X = \{x_1, x_2, ..., x_N\}$ denota el conjunto de vectores de posición de los puntos de captura. La memoria global del pescador es definida como g_{best} (es decir, la mejor solución encontrada entre todos los puntos de captura).
A partir de entonces en FSP se realiza una trayectoria de pesca que abarca cada uno de los puntos de captura (es decir, el pescador parte del punto x_1 hasta el x_2 y así consecutivamente hasta el último punto x_N).

A similitud de un pescador humano, en cada uno de estos puntos de captura, FSP lanza su red de pesca L veces. Esta red de pesca está compuesta por un conjunto de vectores de posición (representando las cuadrículas de la malla de pesca tradicional), $y_{ij} \in Y$, $Y = \{y_{i1}, y_{i2}, ..., y_{iM}\}$, generados dinámicamente a partir del punto de captura x_i correspondiente (donde se encuentre el pescador en ese momento), donde y_{ij} denota el j-ésimo vector en el i-ésimo punto de captura, $i = 1,\ 2, ...,\ N$; $j = 1,\ 2, ...,\ M$. Esta red de pesca está modelada por el algoritmo con el símbolo M (de malla), el cual representa la cantidad de vectores de posición de la red de pesca que serán generados en la vecindad de un punto de captura.

La expresión usada para crear los vectores de posición de esta malla es la siguiente:

$$y_{ij} = x_i + A_j, \tag{5-1}$$

donde x_i es el punto de captura desde donde se va a lanzar la malla y A_j es un vector n-dimensional compuesto por números aleatorios en el rango [-c,c], siendo c un número real denominado coeficiente de amplitud.

El valor de este coeficiente de amplitud es el que define qué vecindad (área) a partir del punto de captura correspondiente cubrirá (explorará) la red de pesca. Para una primera iteración todas las mallas serán generadas con un mismo valor de c, luego este valor puede cambiar condicionado al estado de la pesca en cada punto de captura.

Para el paso siguiente, los vectores de posición de la malla son evaluados y si alguno de estos logra un valor de aptitud (*fitness*) superior que el valor p_i del punto de captura a partir del cual se lanzó esta red de pesca, entonces la referencia que tiene el pescador de este determinado punto de captura es actualizada con esta nueva posición y la malla se redefine considerando este nuevo vector de referencia. Este proceso de actualización del punto de captura, se asemeja al comportamiento del pescador analizado, cuando dentro de un área específica percibe la necesidad de moverse hacia una nueva dirección más prometedora (en la Figura 5-1 de x_1 a x_1'). Cuando el pescador finaliza la pesca en este punto de captura, se actualiza el valor de p_i para el i-ésimo punto de captura, si el *fitness* de p_i es mejor que el del g_{best}, este último es también actualizado.

Durante los lanzamientos que se hagan en un determinado punto de pesca, si el valor de la función de aptitud en p_i para este i-ésimo punto de captura mejora con respecto al lanzamiento anterior, recomendamos disminuir gradualmente el valor del coeficiente de amplitud (por ejemplo, multiplicar c por un factor de 0,95), con lo cual la malla quedará más compacta y en el caso de que la solución esté cerca, reducimos el riesgo de que se nos escape un óptimo local o el global.

Dado el caso contrario, que el valor de la función de aptitud en p_i para el i-ésimo punto de captura no mejora, el valor del coeficiente de amplitud puede ser aumentado con la idea de acelerar la exploración de esta zona de pesca. En este último caso, si después de

algunos lanzamientos encontramos una mejor solución que la registrada en p_i para este punto de captura, el valor del coeficiente de amplitud retoma su valor inicial.

Para completar la ejecución de FSP (al igual que lo haría el pescador), toda esta trayectoria de pesca que se lleva a cabo transitando por cada uno de los puntos de captura se repite T veces.

La secuencia algorítmica de FSP se muestra en el Algoritmo 5.1.

Algoritmo 5.1: Algoritmo FSP

```
Input: T, N, L y M
for i = 1,...,N do
    Inicializar aleatoriamente x_i;
    Evaluar x_i;
    Actualizar p_i;
    Actualizar g_best;
end
for k = 1,...,T do
    for i = 1,...,N do
        while (L > 0) y (se encuentren mejoras) do
            for j = 1,...,M do
                Actualizar y_ij con la expresión y_ij = x_i + A_j;
                Evaluar y_ij;
            end
            Actualizar x_i y p_i;
            L = L − 1;
            Aplicar estrategia para actualizar coeficiente de amplitud, c;
        end
        Actualizar g_best;
    end
end
Result: g_best
```

Ventajas del algoritmo: Finalmente, podemos concluir que FSP es una metaheurística basada en poblaciones de las llamadas *P-Metaheuristics* (*Population based metaheuristics*), pues en el proceso de búsqueda considera múltiples puntos en el espacio que evolucionan en paralelo, y que comparten un objetivo y una memoria común. Las principales ventajas de FSP son:

- Fácil implementación.

- Proporciona una descripción explícita de la idea y el modelo basado en la propia concepción de la metaheurística.
- Por su formulación puede ser aplicado a un gran número de problemas de optimización, así como a los que surgen en situaciones del mundo real en diferentes áreas de ciencias aplicadas, ingeniería y economía.

Luego de la fase de concepción e implementación, FSP también fue guiado por una etapa de parametrización y evaluación con funciones referenciadas y *test* estándares, para demostrar su capacidad para encontrar la solución global del problema, o al menos, soluciones suficientemente próximas. En las siguientes secciones se describen detalladamente estas dos fases.

5.4. Parametrización

En la fase de parametrización, cada parámetro de la metaheurística es evaluado y analizado. Con este objetivo, estudiamos el comportamiento de los parámetros del algoritmo considerando las cinco funciones de prueba de De Jong [118], las cuales han sido ampliamente usadas con este propósito.

La configuración de estas funciones exhibe un balance entre complejidad y diversidad, lo cual resulta necesario en el estudio de una nueva heurística. La dimensión n, la región de búsqueda S, y el mínimo *min* de cada una de estas funciones puede ser observado en la Tabla 5-2 y sus representaciones gráficas en la Figura 5-2.

Las funciones f_1 y f_2, conocidas como las funciones *Sphere* y *Rosenbrock*, respectivamente, son funciones unimodales convexas y bidimensionales. La función f_3, denominada *Step function*, es discontinua, convexa y presenta un único mínimo. Por su parte, la función con ruido gaussiano (f_4) es una función unimodal ruidosa y convexa. Finalmente, la función f_5, referenciada como *Shekel's foxholes*, es una función continua, no convexa, multidimensional y no cuadrática.

Nuestro algoritmo usa cinco parámetros independientes: el número de iteraciones, T; los puntos de captura, N; la cantidad de lanzamientos, L; la cantidad de vectores de posición

Funciones	n	S	**min**
$f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2$	2	$[-500, 500]^n$	0
$f_2\left(\vec{x}\right) = \sum_{i=1}^{n} 100\left(x_i^2 - x_{i+1}\right)^2 + \left(1 - x_i^2\right)^2$	2	$[-5{,}12, 5{,}12]^n$	0
$f_3\left(\vec{x}\right) = \sum_{i=1}^{n} \left(\lfloor x_i + 0{,}5 \rfloor\right)^2$	2	$[-5{,}12, 5{,}12]^n$	0
$f_4\left(\vec{x}\right) = \sum_{i=1}^{n} i\left(x_i^4\right) + Gauss\left(0, 1\right)$	2	$[-1{,}28, 1{,}28]^n$	0
$f_5\left(\vec{x}\right) = \frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} \left(x_i - a_{ij}\right)^6}$	2	$[-65{,}536, 65{,}536]^n$	≈ 1
$a_{ij} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & \ldots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & -16 & \ldots & 32 & 32 & 32 \end{pmatrix}$			

Tabla 5-2: Ecuaciones y atributos de las cinco funciones de prueba de De Jong.

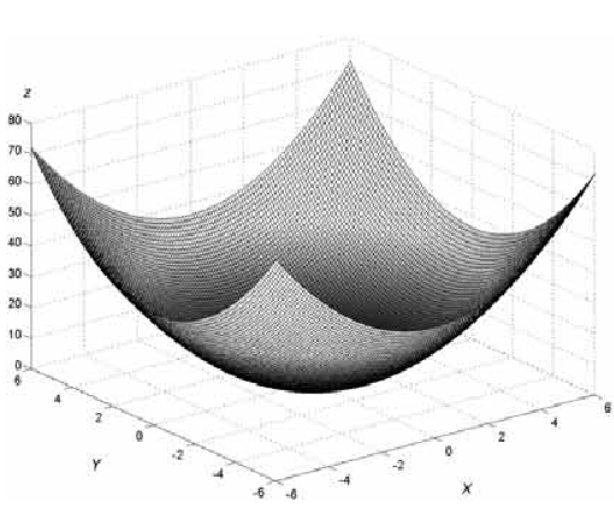
M, que constituyen la red de pesca, y un coeficiente de amplitud c. A continuación realizamos diversas pruebas empíricas para determinar en qué rango de valores tales parámetros alcanzan sus mejores rendimientos frente a este grupo de funciones de referencia.

5.4.1. Puntos de captura

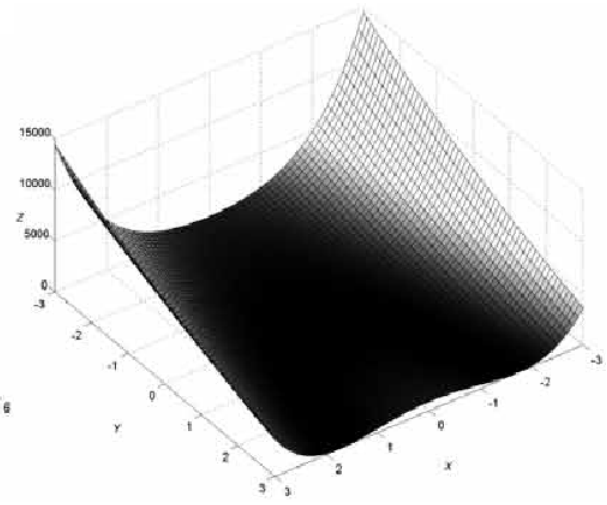
La definición del número de puntos de captura es un factor importante en la estabilidad de nuestra metaheurística. Por ello, realizamos varias pruebas al algoritmo considerando un rango de valores desde 1 hasta 100 puntos de captura. De los resultados obtenidos (ver detalles en la Figura 5-3 con relación a las cinco funciones de De Jong), podemos concluir que para valores de 80 a 100 puntos de captura la heurística logra el comportamiento esperado al encontrar en casi todos los casos, el valor mínimo de la función evaluada. Por tales razones, recomendamos este rango de valores para dicho parámetro.

5.4.2. Número de iteraciones y lanzamientos

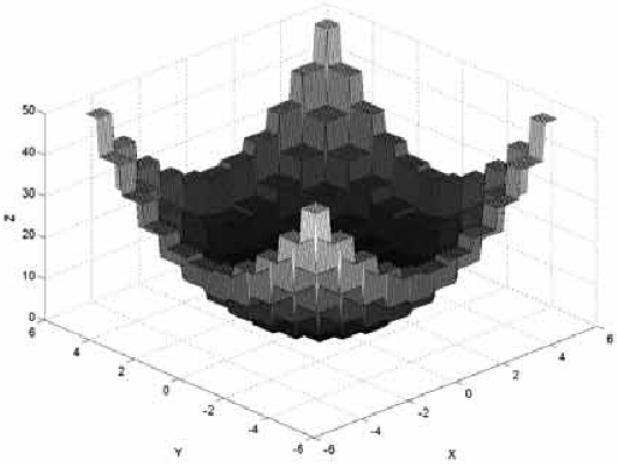
El número de iteraciones y la cantidad de lanzamientos de la red de pesca, son dos parámetros que marcan directamente cuan extensa puede resultar la búsqueda. Un número pequeño de iteraciones y lanzamientos nos lleva a una búsqueda rápida y poco efectiva, sin embargo, fijar valores altos para dichos parámetros nos puede conducir a largos tiempos de respuesta de la heurística. Lo ideal es encontrar un balance en la combinación de estas variables, de tal modo que el algoritmo siempre encuentre resultados deseables, en tiempos



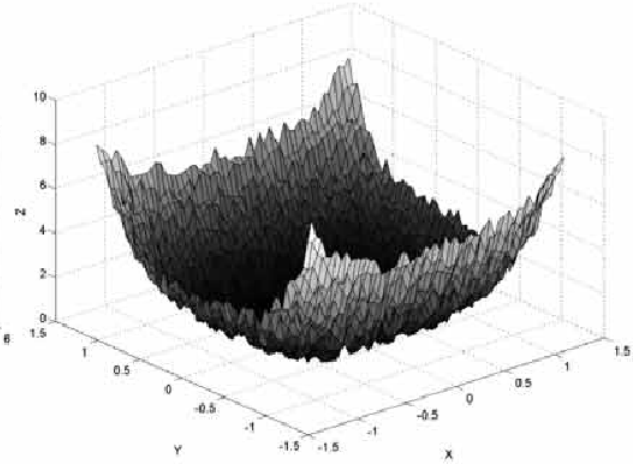
(a) f_1: Sphere Model



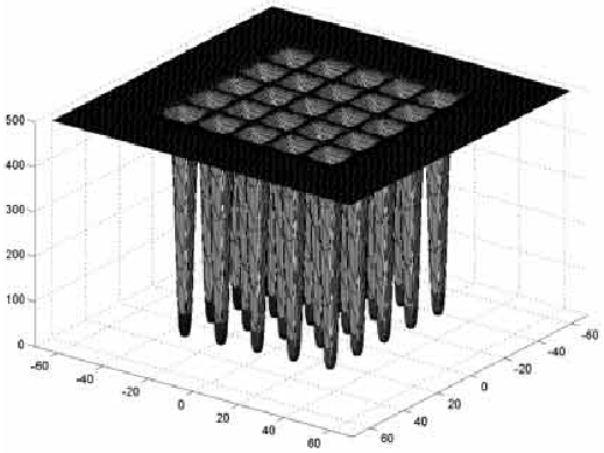
(b) f_2:Rosenbrock's Function



(c) f_3: Step Function



(d) f_4:Function with Gaussian Noise



(e) f_5: Shekel's Foxholes

Figura 5-2: Representación gráfica de las cinco funciones de prueba de De Jong.

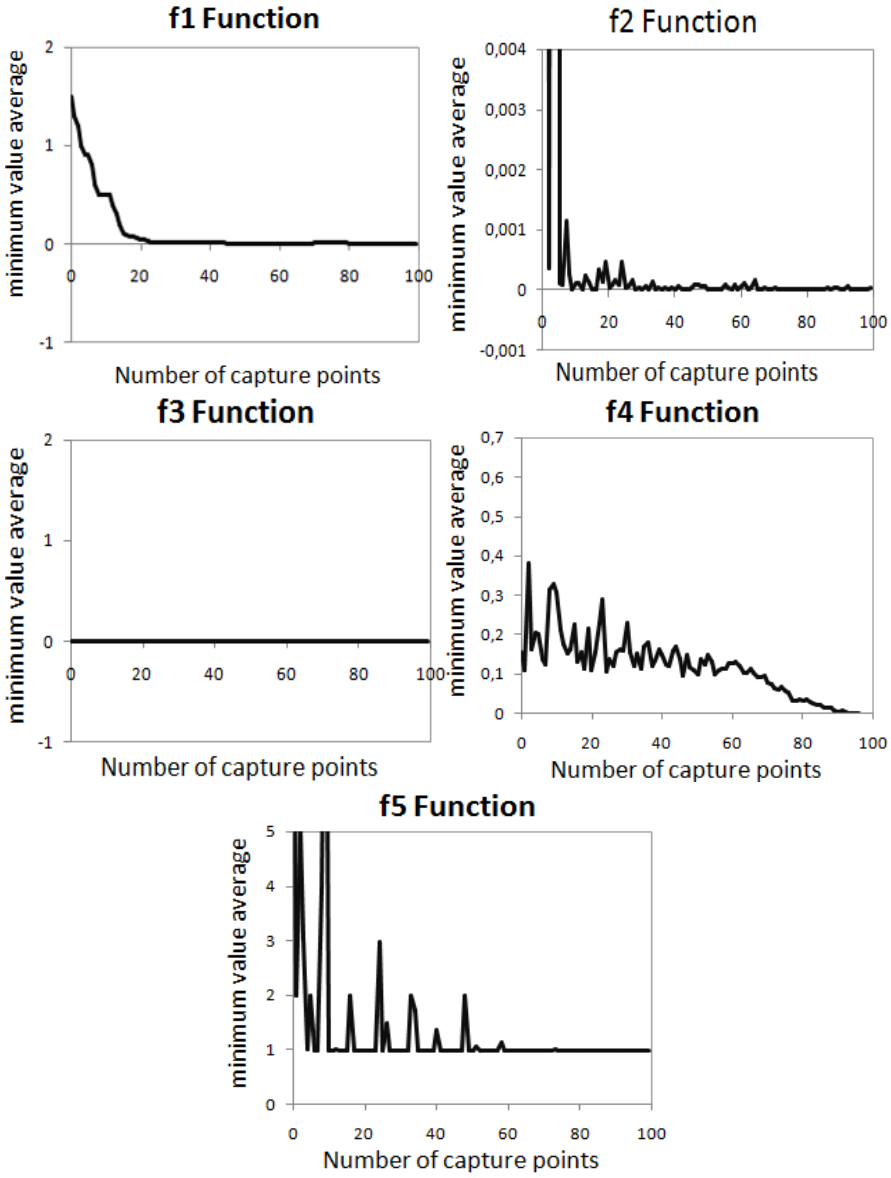


Figura 5-3: Resultados obtenidos en las funciones de De Jong, variando la cantidad de puntos de captura.

razonables.

En la Tabla 5-3 podemos ver los resultados promedio que la heurística alcanza sobre las funciones de De Jong para 30 ejecuciones del algoritmo y considerando varias combinaciones de los parámetros analizados.

En los valores correspondientes a las funciones f_1 y f_2, observamos como el aumento del número de iteraciones mejora gradualmente la precisión del algoritmo para encontrar el

valor óptimo. En ambos casos, se percibe además, que con cinco lanzamientos realizados, el algoritmo exhibe un rendimiento satisfactorio y estable.

Para el caso de la función f_3, el algoritmo siempre alcanza el óptimo (valor mínimo) de la función, desde la primera combinación de estos dos parámetros. Sin embargo, en la función f_4, el aumento de la cantidad de iteraciones y lanzamientos no propicia una mejora con respecto a los valores obtenidos inicialmente.

Finalmente, frente a la función f_5, el algoritmo alcanza óptimos resultados independientemente de los valores asignados a T y L.

Todos estos resultados indican que para estos tipos de funciones en general, FSP logra encontrar el óptimo o una solución lo suficientemente cercana, considerando 5 lanzamientos y ejecutando como promedio un total de 100 iteraciones.

5.4.3. Red de pesca y coeficiente de amplitud

Otro de los elementos claves para que el algoritmo logre un comportamiento eficiente frente a cualquier problema es saber definir adecuadamente cuán compacta y grande será la red de pesca y poder idear estrategias para reajustarla. Los parámetros cantidad de vectores de posición de la malla M y el coeficiente de amplitud c son los que establecen directamente estas características a la red de pesca.

Si tenemos una red de pesca muy densa (de muchos vectores de posición), es decir, de un valor M grande, puede que nos garantice una explotación profunda en la zona de pesca pero aumenta considerablemente la carga computacional, ya que tiene que hacer más llamadas a la función objetivo. Para el caso contrario, en que M fuese muy pequeño, la red de pesca quedaría tan abierta que podríamos dejar escapar aquellas soluciones óptimas. Por otro lado, y en relación a esto, si tenemos un coeficiente de amplitud c muy pequeño, significa que la red de pesca es de poca amplitud (abarca poco espacio) y la búsqueda puede ser lenta y costosa. Sin embargo, para un valor de c muy grande, tendríamos una red de pesca inmensa, que no facilitaría la explotación en el punto de pesca donde nos encontramos. En este sentido, es evidente que para lograr una buena combinación entre estos dos parámetros resulta prudente analizar el problema y llegar a un consenso productivo, siempre considerando que la función de la red de pesca es la de explotar prudencialmente

f	Lanzamientos	Iteraciones				
		100	**200**	**300**	**400**	**500**
f_1	3	1,11 x 10^{-3}	1,52 x 10^{-3}	1,00 x 10^{-3}	8,18 x 10^{-4}	3,15 x 10^{-4}
	5	1,10 x 10^{-4}	5,27 x 10^{-4}	4,12 x 10^{-5}	8,36 x 10^{-6}	4,25 x 10^{-7}
	10	9,21 x 10^{-4}	8,97 x 10^{-4}	6,54 x 10^{-5}	5,66 x 10^{-6}	6,54 x 10^{-7}
	15	5,25 x 10^{-4}	6,54 x 10^{-5}	7,58 x 10^{-4}	6,58 x 10^{-6}	3,35 x 10^{-6}
	20	3,15 x 10^{-4}	7,45 x 10^{-5}	2,22 x 10^{-5}	1,01 x 10^{-5}	9,02 x 10^{-7}
	25	4,96 x 10^{-4}	9,87 x 10^{-4}	6,35 x 10^{-4}	5,47 x 10^{-5}	2,55 x 10^{-6}
f_2	3	4,16 x 10^{-6}	9,99 x 10^{-6}	8,15 x 10^{-6}	1,29 x 10^{-7}	5,65 x 10^{-7}
	5	1,36 x 10^{-7}	4,55 x 10^{-7}	7,33 x 10^{-7}	8,24 x 10^{-8}	4,27 x 10^{-8}
	10	5,42 x 10^{-6}	6,49 x 10^{-7}	6,32 x 10^{-6}	7,69 x 10^{-6}	8,57 x 10^{-7}
	15	3,25 x 10^{-7}	6,54 x 10^{-7}	4,25 x 10^{-7}	4,36 x 10^{-8}	8,73 x 10^{-8}
	20	6,14 x 10^{-7}	2,11 x 10^{-7}	9,68 x 10^{-7}	9,58 x 10^{-8}	1,68 x 10^{-8}
	25	1,20 x 10^{-6}	3,48 x 10^{-6}	4,57 x 10^{-6}	4,77 x 10^{-7}	9,47 x 10^{-7}
f_3	3	0	0	0	0	0
	5	0	0	0	0	0
	10	0	0	0	0	0
	15	0	0	0	0	0
	20	0	0	0	0	0
	25	0	0	0	0	0
f_4	3	1,27 x 10^{-2}	6,74 x 10^{-3}	9,00 x 10^{-2}	5,44 x 10^{-3}	7,82 x 10^{-3}
	5	3,55 x 10^{-3}	4,65 x 10^{-2}	7,33 x 10^{-3}	1,04 x 10^{-2}	8,14 x 10^{-3}
	10	9,58 x 10^{-2}	2,11 x 10^{-3}	1,54 x 10^{-2}	2,62 x 10^{-3}	1,34 x 10^{-2}
	15	1,82 x 10^{-2}	7,05 x 10^{-2}	9,87 x 10^{-3}	1,87 x 10^{-2}	1,24 x 10^{-3}
	20	5,68 x 10^{-3}	9,88 x 10^{-2}	3,77 x 10^{-3}	9,52 x 10^{-3}	1,99 x 10^{-2}
	25	3,65 x 10^{-3}	8,47 x 10^{-2}	2,03 x 10^{-2}	2,41 x 10^{-3}	7,44 x 10^{-3}
f_5	3	1,0582	0,9980	0,9980	0,9980	0,9980
	5	0,9980	0,9980	0,9980	0,9980	0,9980
	10	0,9980	0,9980	0,9980	0,9980	0,9980
	15	0,9980	0,9980	0,9980	0,9980	0,9980
	20	0,9980	0,9980	0,9980	0,9980	0,9980
	25	0,9980	0,9980	0,9980	0,9980	0,9980

Tabla 5-3: Resultados promedios obtenidos para diferentes números de iteraciones y lanzamientos.

las vecindades de los puntos de pesca.

FSP en un principio trata parte de este dilema aplicando una estrategia sencilla (descrita en la Sección 5.2.) para actualizar el coeficiente de amplitud. A pesar de esto, la selección

de c y M está condicionada como la mayoría de los parámetros de las heurísticas, según el problema, al tamaño de la región de búsqueda, etc.

En la Tabla 5-4, se presenta en resumen no sólo de los mejores valores para los parámetros c y M, sino además de todos los valores recomendados para cada uno de los parámetros de FSP, en relación empírica a rendimientos óptimos frente a las funciones de prueba de De Jong.

Además, si analizamos la región de búsqueda S correspondiente a cada función de prueba, podemos apreciar que los valores de c tienden a incrementar en aquellas regiones más grandes, no siendo así para el caso de M.

Funciones	**Parámetros**				
	T	**N**	**L**	**M**	**c**
f_1	500	25	5	200	20,0
f_2	400	80	5	80	0,10
f_3	100	10	3	30	0,70
f_4	100	100	3	100	0,05
f_5	100	80	5	80	10,0

Tabla 5-4: Parámetros recomendados para FSP según su evaluación ante las funciones de prueba de De Jong.

Finalmente, sería oportuno tener presente que estas recomendaciones son hechas en relación a las funciones de prueba utilizadas, pero resulta evidente que para otros problemas y situaciones totalmente diferentes, habría que volver a ajustar empíricamente los valores de cada parámetro.

5.5. Experimentación

En esta sección, presentamos los resultados experimentales de la fase de evaluación de nuestra propuesta en términos de ejecuciones satisfactorias (convergencia en valores mínimos globales) y consumo de tiempo. Se llevaron a cabo comparaciones considerando otras metaheurísticas representativas y referenciadas en la literatura. Fueron usadas en la eva-

luación un conjunto de funciones de referencia (*benchmark functions*) ampliamente usadas para probar el comportamiento y la convergencia de métodos heurísticos. Finalmente, se muestran y analizan los resultados alcanzados por los distintos métodos.

5.5.1. Funciones estándares para las pruebas

Con el fin de evaluar nuestro método FSP, usamos un conjunto de funciones de prueba propuestas por Molga y Smutnicki [120]. Estas funciones pueden ser consultadas en la Tabla 5-5, donde se muestra también información sobre sus dimensionalidades (n), espacios de búsqueda ($S \subset R^n$) y los valores para los cuales alcanzan el mínimo global (f_{min}).

Este conjunto de problemas es muy diverso, pues incluye indistintamente funciones con variables correlacionadas y no correlacionadas, unimodales y multimodales. Por ejemplo, las funciones f_1-f_3 son unimodales y las funciones f_4 y f_5 son funciones multimodales donde el número de mínimos locales aumenta exponencialmente con la dimensión del problema. Por su parte, las funciones f_6-f_{11} son funciones de baja dimensionalidad, que tienen sólo algunos mínimos locales.

5.5.2. Metaheurísticas seleccionadas para la comparación

Con la intención de comparar nuestra propuesta con otros procedimientos heurísticos referenciados en la literatura, seleccionamos los siguientes métodos: procedimientos de búsqueda basados en funciones voraces aleatorizadas que se adaptan (GRASP, *Greedy Randomized Adaptive Search Procedures*) [121], optimización con enjambre de partículas (PSO, *Particle Swarm Optimization*) [96] [122] [98], y evolución diferencial (DE, *Differential Evolution*) [123]. Esta selección fue realizada también con el objetivo de evaluar el comportamiento de FSP, tanto con algoritmos que siguen estrategias similares (GRASP), como con otros completamente diferentes (PSO y DE).

GRASP [121] es una metaheurística multi-arranque en la que cada iteración consiste básicamente de dos etapas: construcción y búsqueda local. Se basa en la premisa de que soluciones iniciales diversas y de buena calidad juegan un papel importante en el éxito de métodos locales de búsqueda. Este procedimiento se repite varias veces y la mejor solución encontrada en todas las iteraciones se devuelve como la solución aproximada. Una de las principales desventajas del GRASP puro es su falta de estructuras de memoria. Las

Functions	n	S	f_{min}
$f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2$	5	$[-5{,}12, 5{,}12]^n$	$f_1(\vec{0}) = 0$
$f_2(\vec{x}) = \sum_{i=1}^{n} (i \cdot x_i^2)$	5	$[-5{,}12, 5{,}12]^n$	$f_2(\vec{0}) = 0$
$f_3(\vec{x}) = \sum_{i=1}^{n} \sum_{j=1}^{i} x_j^2$	2	$[-65{,}536, 65{,}536]^n$	$f_3(\vec{0}) = 0$
$f_4(\vec{x}) = \sum_{i=1}^{n} \left[-x_i \sin\left(\sqrt{\lvert x_i \rvert}\right) \right]$	2	$[-500, 500]^n$	$f_4(\vec{420{,}9687}) = -418{,}9829n$
$f_5(\vec{x}) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$	2	$[-1{,}0, 1{,}0]^n$	$f_5(\vec{0}) = 0$
$f_6(x_1, x_2) = \left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$	2	$[-1{,}9, 2{,}0]$	$f_6(1, 1) = 0$
$f_7(x, y) = (x - 13 + ((5 - y)\, y - 2)\, y)^2 + (x - 29 + ((y + 1)\, y - 14)\, y)^2$	2	$[-1, 12]$	$f_7(5, 4) = 0$, $f_7(11{,}41, -0{,}8986) = 48{,}9842$
$f_8(x_1, x_2) = a\left(x_2 - bx_1^2 + cx_1 - d\right)^2 + e\,(1 - f)\cos(x_1) + e$	2	$[-\pi, 13]$	$f_8(-\pi, 12{,}275) = 0{,}397887$, $f_8(\pi, 2{,}275) = 0{,}397887$ $f_8(9{,}42478, 2{,}475) = 0{,}397887$
$f_9(x, y) = (1{,}5 - x\,(1 - y))^2 + \left(2{,}25 - x\left(1 - y^2\right)\right)^2 + \left(2{,}625 - x\left(1 - y^3\right)\right)^2$	2	$[0{,}0, 5{,}0]$	$f_9(3, 0{,}5) = 0$
$f_{10}(x_1, x_2) = \left(4 - 2{,}1x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1 x_2 + \left(-4 + 4x_2^2\right) x_2^2$	2	$x_1[-3, 3]$, $x_2[-2, 2]$	$f_{10}(-0{,}0898, 0{,}7126) = -1{,}0316$, $f_{10}(0{,}0898, -0{,}7126) = -1{,}0316$
$f_{11}(x_1, x_2) = \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \cdot \left[30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$	2	$[-2, 2]$	$f_11\,(0, -1) = 3$

Tabla 5-5: Funciones de referencia para evaluar métodos heurísticos.

iteraciones de GRASP son independientes y no utilizan las observaciones hechas durante iteraciones previas.

Por su parte, PSO es una técnica de optimización estocástica basada en poblaciones desarrollada por Eberhart y Kennedy en 1995 [96], e inspirada en el comportamiento social observado en animales o insectos. En la Sección 4.2 del capítulo 4 se describe el algoritmo clásico de PSO. Las principales ventajas de este método son: implementación simple, fácilmente paralelizable para procesos concurrentes, libre de derivadas, consta de muy pocos parámetros, y ha demostrado ser un algoritmo de búsqueda global muy eficiente en ambientes complejos. A pesar de esto, presenta la desventaja de tener una convergencia lenta en la etapa de refinación de la búsqueda (débil capacidad de búsqueda local).

El algoritmo DE creado por Storn y Price [123] es un método de optimización perteneciente a la categoría de computación evolutiva. Este método mantiene una población de soluciones candidatas a las que se les aplica un procedimiento de mutación y recombinación para generar nuevos individuos. Los nuevos individuos se eligen de acuerdo al valor de su función de fitness. La principal característica de DE es el uso de vectores de prueba que compiten con los individuos de la población actual con el fin de sobrevivir. Una de sus mayores ventajas es que supera los inconvenientes innatos de los algoritmos genéticos: convergencia prematura y la falta de habilidad para desarrollar una buena búsqueda local. Sin embargo, DE a veces tiene una capacidad limitada para mover sus poblaciones a través de grandes distancias por el espacio de búsqueda y esto le puede provocar estados de estancamiento.

Aunque FSP no siempre supera la actuación de otros métodos, este si resuelve alguno de sus inconvenientes, por ejemplo, la independencia en las iteraciones de GRASP, la débil capacidad de búsqueda local del PSO, y la habilidad limitada de DE para salir de zonas estancadas.

5.5.3. Evaluación de rendimientos

Como habíamos mencionado anteriormente, los algoritmos seleccionados para la comparación fueron GRASP, DE y PSO, este último incluyendo peso inercial y factor de constricción. Durante el estudio comparativo, estos métodos serán referenciados con las siguientes

etiquetas: “GRASP”, “DE” y “PSO-w-cf”.

En todos los experimentos y evaluaciones los algoritmos fueron ejecutados 30 veces para cada función de prueba. La población inicial fue generada de manera uniforme y aleatoria en los espacios de búsqueda correspondientes a cada función de prueba (véase Tabla 5-5). En el caso de los parámetros (por ejemplo, el número de iteraciones) que influyen en el número de evaluaciones que realizarán los algoritmos, tratamos de ser consecuentes poniendo el mismo valor a todos los métodos para buscar una mayor igualdad de condiciones a la hora de realizar las comparaciones.

Específicamente los valores asignados a los parámetros de GRASP fueron los siguientes: 100 iteraciones, 150 elementos compondrán la lista restringida de candidatos, el criterio de parada de la búsqueda local está condicionado a no encontrar mejoras en la vecindad de la solución actual, y se utilizó la estrategia *first-improving* que consistió en mover la solución actual hacia el primer vecino cuyo valor del costo de la función de fitness es menor que el de la solución actual.

Los valores utilizados para los parámetros de PSO-w-cf fueron: 100 iteraciones, 80 partículas, el peso inercial w fue fijado al valor 0,9 y se controló su decremento hasta 0,4. Los vectores n-dimensionales n_1 y n_2 fueron instanciados por números aleatorios en el intervalo [0,1]. Las constantes de aceleración c_1 (*particle increment*) y c_2 (*global increment*) fueron ambas instanciadas en 2,05. El factor de constricción $\chi = 0{,}729$, y como velocidad máxima V_{max} el 50 % del valor real que se corresponde con la amplitud del rango numérico que puede ocupar la variable en cada dimensión. La configuración de estos parámetros fue realizada considerando algunos trabajos referenciados [97, 107] sobre el rendimiento y análisis del comportamiento de PSO.

Para el caso de DE se utilizaron: 100 iteraciones, 80 vectores iniciales, 6 como número máximo de vectores objetivos, el parámetro que controla la tasa de mutación $f = 0{,}75$ y para el control de la recombinación un $cr = 0{,}5$. Estos dos últimos parámetros fueron fijados considerando las pruebas empíricas realizadas originalmente por Storn y Price [123].

Los parámetros de nuestro método FSP fueron instanciados con los siguientes valores: 100 iteraciones, 80 puntos de captura, 5 lanzamientos, una red de pesca compuesta por 80

Función	**Algoritmo**	**Mejor**	**Media**	**Tiempo**	**Ejec.Satisf.**
	DE	1,7328e-02	9,5487e-02	2,58	05
f_1	PSO-w-cf	8,3241e-13	3,6998e-11	**0,98**	**30**
(De Jong's)	GRASP	1,4805e-01	1,17728420	76,1	00
	FSP	1,4058e-02	7,7205e-02	2,21	07
	DE	1,6242e-02	8,8914e-02	9,21	04
f_2	PSO-w-cf	6,1828e-10	9,1778e-09	**1,04**	**30**
(Axis parallel	GRASP	0,42167256	1,67167256	85,3	00
hyper-ellipsoid)	FSP	1,7272e-02	9,3354e-02	9,91	02
	DE	6,1984e-05	7,1452e-05	0,06	**30**
f_3	PSO-w-cf	1,2611e-04	2,2155e-04	**0,03**	**30**
(Rotated	GRASP	4,1835e-03	3,1987e-02	1,51	00
hyper-ellipsoid)	FSP	1,7315e-03	1,7956e-02	0,06	08
	DE	-837,96586	-837,36187	**0,06**	02
f_4	PSO-w-cf	-837,96586	-837,12575	0,08	05
(Schwefel's)	GRASP	-837,91124	-837,21545	7,74	00
	FSP	-837,96581	-837,61520	0,08	**06**
	DE	1,7257e-05	8,5222e-04	**0,01**	18
f_5	PSO-w-cf	1,9099e-05	6,6587e-04	**0,01**	21
(Sum of different	GRASP	2,9311e-04	3,1533e-03	0,06	00
power functions)	FSP	8,6995e-06	1,9112e-05	**0,01**	**30**
	DE	8,4157e-03	2,4723e-02	**0,01**	00
f_6	PSO-w-cf	1,6448e-05	5,3687e-04	**0,01**	**16**
(Rosenbrock	GRASP	5,7065e-04	1,2365e-03	0,05	00
or Banana)	FSP	1,5161e-05	2,8734e-04	**0,01**	14
	DE	5,1626e-04	1,1234e-03	0,07	**30**
f_7	PSO-w-cf	1,0961e-04	3,5698e-03	**0,06**	23
(Freudenstein	GRASP	1,9228e-03	1,4589e-02	1,09	11
and Roth)	FSP	3,0940e-03	1,1090e-03	**0,06**	**30**
	DE	0,39781959	0,40254578	**0,03**	01
f_8	PSO-w-cf	0,39788737	0,39998751	**0,03**	**25**
(Branins's)	GRASP	0,39782849	0,41587485	0,67	01
	FSP	0,39787495	0,39954781	**0,03**	12
	DE	1,3211e-03	3,6587e-02	**0,03**	18
f_9	PSO-w-cf	1,8051e-05	2,7624e-04	**0,03**	**30**
(Beale)	GRASP	3,3778e-04	9,5874e-03	1,07	20
	FSP	5,5255e-04	1,4423e-03	**0,03**	**30**
	DE	-1,0316284	-1,0375987	**0,04**	**08**
f_{10}	PSO-w-cf	-1,0316625	-1,0335411	**0,04**	01
(Six-hump	GRASP	-1,0310172	-1,0405474	0,99	00
camel back)	FSP	-1,0316738	-1,0381120	**0,04**	02
	DE	3,00000082	3,00054785	**0,03**	**30**
f_{11}	PSO-w-cf	3,00711378	3,01965874	**0,03**	21
(Goldstein-Price's)	GRASP	3,00664548	3,09925453	0,71	17
	FSP	3,00285013	3,01125478	**0,03**	23

Tabla 5-6: Rendimientos alcanzados por los algoritmos frente a cada función de prueba.

vectores de posición, y un coeficiente de amplitud que fue ajustado a 0,05.

Los resultados de todas estas evaluaciones se muestran en la Tabla 5-6, donde la primera columna hace referencia a las funciones de prueba utilizadas, la segunda a los algoritmos evaluados, la tercera al mejor valor alcanzado por cada algoritmo en relación al valor de la función cuando alcanza el mínimo global, la cuarta al valor medio de cada algoritmo considerando sus 30 ejecuciones para cada función, la sexta al tiempo promedio (en segundos) consumido por cada algoritmo para dar solución al problema, y la séptima columna (etiquetada como "Ejec.Satisf.") representa la cantidad de veces que el algoritmo llega a la solución deseada con un nivel de precisión de 1e-02. Todas las evaluaciones fueron realizadas sobre una arquitectura Intel(R) Core(TM) i5 (4 CPUs) 2,53GHz y 8,0 GBytes de RAM.

En estos resultados podemos observar como FSP para las funciones f_5, f_7 y f_9 siempre converge (en todas sus ejecuciones) a la solución deseada y con el nivel de precisión establecido. Para las funciones f_4 y f_{10} donde se registran los peores rendimientos de todos los algoritmos, FSP se sitúa en la primera y segunda mejor posición con respecto a los demás métodos.

En general, podemos concluir que FSP en el 90 % de los casos y comparaciones, se ubica entre los dos mejores resultados en cuanto a ejecuciones satisfactorias, y en el otro 10 % siempre encuentra soluciones óptimas y nunca queda en último lugar.

Por otro lado, con relación al tiempo consumido en el procesamiento, FSP muestra resultados similares a PSO y DE, logrando para un 82 % de las funciones de prueba tener los mejores o segundo mejores tiempos de cómputo.

Finalmente, todos estos resultados demuestra las potencialidades de FSP en la solución de problemas de optimización.

5.6. Conclusiones

En este capítulo se desarrolló y validó una nueva propuesta metaheurística para tratar problemas de optimización global. Este método denominado Procedimiento de Búsqueda del Pescador está basado en el comportamiento y pericias observadas en la labor habitual de un pescador tradicional. Específicamente, este algoritmo heurístico está concebido para

encontrar soluciones eficientes siguiendo una estrategia combinada de búsqueda guiada y búsqueda local.

Además se llevó a cabo una fase de parametrización considerando las cinco funciones de prueba de De Jong, para analizar y determinar sobre que rango de valores los parámetros manifestaban mejores rendimientos.

También, se realizaron una serie de pruebas de rendimiento a FSP utilizando un conjunto de funciones estándares y de referencia para este fin. Durante esta fase se comparó el rendimiento, la capacidad de convergencia a valores mínimos globales y el coste de tiempo de nuestro método frente a tres métodos heurísticos bien conocidos: PSO, GRASP y DE. Los resultados obtenidos mostraron como FSP en el 90 % de las comparaciones, se sitúa entre los dos mejores resultados en cuanto a ejecuciones satisfactorias, y en el resto (10 %) de las pruebas mantiene un comportamiento promedio, siempre encontrando soluciones óptimas. Por otro lado, FSP muestra resultados similares a PSO y DE en cuanto al tiempo de respuesta del algoritmo (tiempo consumido en el procesamiento), alcanzando el mejor y segundo mejor tiempo de cómputo para el 82 % de las funciones evaluadas.

Finalmente, podemos decir que FSP muestra ser una metaheurística simple, intuitiva, que por su formulación puede ser aplicada a un gran número de problemas de optimización y que manifiesta un comportamiento eficiente en la búsqueda de soluciones para todos los problemas teóricos evaluados y en comparación con otras propuestas heurísticas referenciadas en la literatura.

Capítulo 6

Aplicación del Procedimiento de Búsqueda del Pescador al Aprendizaje de la Ordenación

"La ciencia es el alma de la prosperidad de las naciones y la fuente de vida de todo progreso".
— Louis Pasteur

6.1. Introducción

En este capítulo presentamos una variante adaptada y mejorada del Procedimiento de Búsqueda del Pescador para tratar el problema del L2R. A este nuevo método lo denominaremos RankFSP y es capaz de construir una función de ordenación optimizando directamente cualquier medida de evaluación según un determinado conjunto de entrenamiento. Además, RankFSP implementa una estrategia para evadir los mínimos locales y los puntos de búsqueda estancados. Con el objetivo de mejorar la tarea del *ranking* que realizan los métodos de L2R sobre las colecciones que serán analizadas, se propone también un método simple de selección de rasgos. Posteriormente, se realizó un estudio experimental donde se evalúo el rendimiento de RankFSP y su efectividad en comparación con otros métodos referenciados en la literatura. Todos los resultados obtenidos fueron analizados estadísticamente. Finalmente, se llevó a cabo un análisis del coste computacional de los métodos propuestos en relación al tiempo de procesamiento y a la cantidad de consultas evaluadas.

6.2. Método RankFSP

En esta sección, introducimos formalmente un nuevo método de L2R para la RI, al cual denominaremos RankFSP. Este algoritmo está basado en el Procedimiento de Búsqueda del Pescador (FSP).

Antes de describir el método RankFSP es conveniente mencionar de qué forma fueron adaptados algunos elementos de FSP para su integración dentro del problema del L2R y qué otras mejoras o modificaciones le fueron incorporadas al algoritmo.

RankFSP fue concebido como un algoritmo que usa una función de *ranking* lineal basada en rasgos (extraídos a partir de un documento sobre la base de una consulta), por lo que la dimensión del vector de posición de un punto de captura queda determinado por este número de rasgos. Cada punto de captura constituye una posible solución, es decir, un vector de pesos que será evaluado sobre el conjunto de entrenamiento utilizando dicha función de *ranking* lineal. La función de *ranking* f que devuelve el algoritmo en su última iteración será construida con el valor finalmente registrado en la memoria global g_{best}. Para una descripción más detallada de la configuración de los parámetros en RankFSP consulte la Sección 6.4.1.

Durante este proceso de adaptación de FSP al problema del L2R y después de haber realizado algunas pruebas empíricas iniciales sobre las colecciones de datos de L2R, decidimos incorporar dos nuevas estrategias a RankFSP para mejorar su rendimiento en el *ranking*. Las cuales pueden ser extendidas e insertadas en el FSP clásico con vista a su aplicación en futuros problemas de optimización. Estas variantes la etiquetaremos como: estrategia de lanzamientos condicionados al contexto y estrategia de reinicios de puntos de captura.

En la primera estrategia, a diferencia de FSP que en cada punto de captura el pescador siempre realiza una cantidad fija de lanzamientos, se plantea que la cantidad de lanzamientos que se realicen en un determinado punto de captura estará condicionada a ir encontrando mejoras con respecto al valor de la función en p_i. Es decir, mientras que se vaya mejorando la solución actual de un punto de captura con cada lanzamiento el pescador continuará lanzando la malla en esta zona hasta que la pesca falle (no encuentre mejoras). Dado dicho caso, el pescador se mueve para el siguiente punto de captura. En esta estrategia, se optimiza y se logra un mejor aprovechamiento de la cantidad de lanzamientos a realizar durante la pesca.

Por su parte, la segunda estrategia consiste en tratar de identificar y evadir aquellos caladeros que lleven al pescador hacia estancamientos en mínimos locales o hacia zonas de búsqueda donde no se encuentren mejoras. Con este objetivo, el algoritmo irá chequeando

el comportamiento (evolución de la pesca) de los puntos de captura y todos aquellos que durante un determinado número de iteraciones no mejore su posición, es decir, no se encontró ninguna buena solución a partir de ellos, serán redefinidos a nuevas posiciones en el espacio de búsqueda. Este reinicio puede ser aleatorio o controlado (guiado hacia zonas de pesca inexploradas). Estas variantes y aportes de RankFSP serán más detalladas y contextualizadas con la propia descripción del algoritmo.

Introduciendo el procedimiento general de RankFSP, tenemos que este algoritmo tiene como parámetros de entrada un conjunto de entrenamiento S, una medida de evaluación E, un número de iteraciones T, un número de puntos de captura N y una red de pesca cuyo tamaño está representado por el valor M.

En los primeros pasos del algoritmo, RankFSP, al igual que FSP, inicializa aleatoriamente un determinado conjunto de puntos de captura (soluciones proyectadas como vectores de pesos), instanciando la memoria local p_i correspondiente a cada punto de captura x_i y actualizando el vector de posición global g_{best} (mejor solución encontrada hasta el momento para el conjunto de entrenamiento).

Comienza entonces una trayectoria de pesca de T rondas, donde el pescador se moverá por cada uno de los puntos de captura. Una vez situado en un determinado punto de captura y como ya habíamos mencionado, se aplicará la estrategia de lanzamientos condicionados al contexto. En consonancia con ello, el pescador se mantendrá en este caladero mientras los lanzamientos de la red de pesca sean efectivos (encuentre mejores soluciones locales). Cuando falle la pesca, procederá a moverse para el siguiente punto de captura.

En cada punto de captura x_i, los vectores de posición y_{ij} de la red de pesca son evaluados con respecto al conjunto de entrenamiento. La función de pérdida $R(f)$ representa la función de aptitud usada para evaluar cada nuevo vector de posición. Esta función de aptitud usa la medida de evaluación E y la predicción π_i, obtenida a partir de la aplicación de la expresión $f(q_i, d_{ij}) = {x_i}^T \phi(q_i, d_{ij})$ sobre el conjunto de entrenamiento S.

Una vez finalizada la pesca en un punto de captura x_i, y antes de pasar al siguiente, RankFSP actualiza (si resulta pertinente) la memoria local p_i y la memoria global g_{best}.

Analicemos un ejemplo para entender mejor la secuencia algorítmica de RankFSP hasta este momento. Para ello vamos a utilizar el mismo conjunto de entrenamiento (ver Tabla 6-1) que sirvió de ejemplo para ilustrar el método RankPSO en la Sección 4.3.1, donde quedó claro que cada documento está representado por dos rasgos extraídos en relación a la consulta formulada y que un especialista asignó una etiqueta de relevancia a cada uno de estos documentos.

Etiqueta de Relevancia	**ID. Consulta**	**Rasgo n^o1**	**Rasgo n^o2**	**ID. Doc.**
1	1	0,712	0,708	D01
0	1	0,633	0,801	D02
2	1	0,750	0,923	D03
0	1	0,490	0,910	D04

Tabla 6-1: Ejemplo sencillo de conjunto de entrenamiento para el L2R.

Apliquemos entonces RankFSP sobre dicho conjunto de datos, considerando tres puntos de captura, una malla compuesta por tres vectores de posición y como medida de evaluación implícita en la función de pérdida $R(f)$ MAP.

Siguiendo la Figura 6-1 tenemos que el pescador inicializó aleatoriamente (con distribución uniforme) tres caladeros en toda la región de pesca (espacio de búsqueda). El primer punto de pesca x_1 fue ubicado en la posición (3,3), el segundo x_2 en la (8,3) y el tercero x_3 en la posición (7,5,8,3). Para evaluar el primer punto de pesca, utilizamos su vector de posición como vector de pesos en la función de *ranking* $f(q_i, d_{ij}) = {x_i}^T \phi(q_i, d_{ij})$, que aplicaremos de la siguiente manera sobre el conjunto de entrenamiento:

$$\begin{pmatrix} f(q_1, D01) = 3*0,712 + 3*0,708 = 4,26 \\ f(q_1, D02) = 3*0,633 + 3*0,801 = 4,302 \\ f(q_1, D03) = 3*0,750 + 3*0,923 = 5,019 \\ f(q_1, D04) = 3*0,490 + 3*0,910 = 4,2 \end{pmatrix} \begin{matrix} \\ ranking \\ descendente \\ \\ \end{matrix} \Rightarrow \begin{pmatrix} D03 \\ D02 \\ D01 \\ D04 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad MAP \begin{pmatrix} 2 \\ 0 \\ 1 \\ 0 \end{pmatrix} = 0,83,$$

donde finalmente la función de pérdida sería calculada como $R(f) = 1 - MAP = 1 - 0,83 = 0,17$. Este valor resultante (0,17) constituye el valor de p_1 (memoria local) en el punto de captura x_1. Este mismo procedimiento fue llevado a cabo para los otros dos puntos de captura, donde finalmente, el mejor de todos fue x_2 con un MAP ideal igual a 1 (con pérdida cero), que resaltamos en el gráfico con un circulo de color rojo.

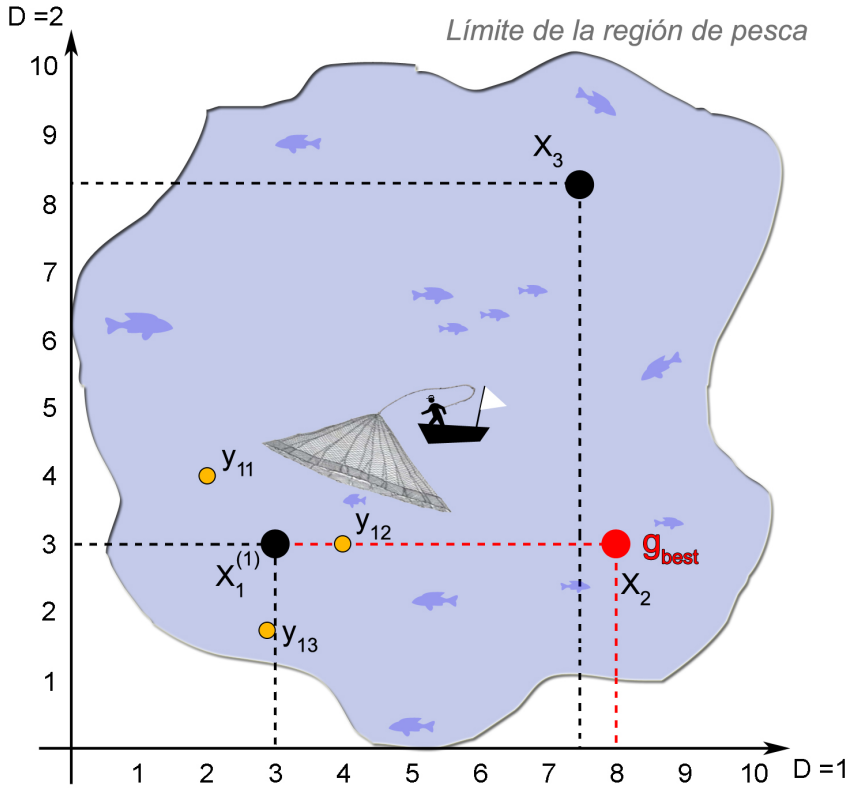


Figura 6-1: Primer lanzamiento de la malla del pescador.

Luego, el pescador situado en el punto de captura x_1 genera una malla de pesca de tres vectores de posición a partir de la expresión $y_{ij} = x_i + A_j$. Donde para este caso, utilizaremos un coeficiente de amplitud $c = 1$, con lo cual, el vector bidimensional A_j tomará valores en el rango [-1,1]. Este proceso es formulado como mostraremos a continuación:

Cálculo del primer vector de posición de la malla, dado un vector aleatorio A(-1;1):

$y_{111} = x_{11}^{(1)} + A_1 = 3 + (-1) = 2,$

$y_{112} = x_{12}^{(1)} + A_2 = 3 + 1 = 4.$

Cálculo del segundo vector de posición de la malla, dado un vector aleatorio A(1;0):

$y_{121} = x_{11}^{(1)} + A_1 = 3 + 1 = 4,$

$y_{122} = x_{12}^{(1)} + A_2 = 3 + 0 = 3.$

Cálculo del tercer vector de posición de la malla, dado un vector aleatorio A(-0,1;-0,3):

$y_{131} = x_{11}^{(1)} + A_1 = 3 + (-0,1) = 2,9,$

$y_{132} = x_{12}^{(1)} + A_2 = 3 + (-0,3) = 2,7.$

Donde en la representación de y_{111} la secuencia de los subíndices hace referencia a: vector generado a partir del punto de captura n^o1, primer vector de la maya y valor de este vector para la primera dimensión.

En el gráfico de la Figura 6-1 podemos observar como quedaron distribuidos tales vectores de la malla (puntos naranjas alrededor del punto de captura x_1).

Estos vectores de posición de la malla son evaluados de la misma forma como evaluamos cada punto de captura, obteniéndose para y_{11} un MAP=0,75, en y_{12} el valor fue 1 (ideal), y en y_{13} se alcanzó un MAP igual a 0,83. Como el valor de la función de aptitud en x_1 es 0,17, y en el punto y_{12} se alcanza una pérdida igual a cero (MAP=1)(la mejor solución), entonces el pescador mueve el punto de captura x_1 a la posición y_{12}, y actualiza su valor de p_1 con esta nueva posición. En la gráfica de la Figura 6-2 se puede apreciar este cambio, y además como el pescador situado en este nuevo vector vuelve nuevamente a lanzar su malla.

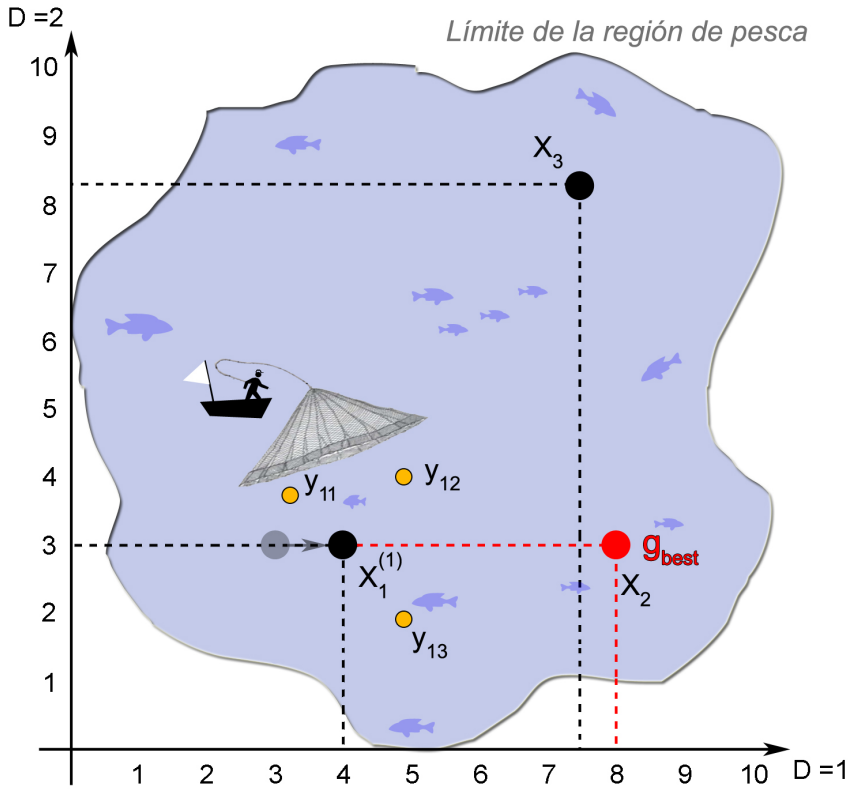


Figura 6-2: Segundo lanzamiento de la malla del pescador.

Una vez que los puntos de la malla no mejoren la solución en x_1 el pescador finaliza la pesca en este punto de captura, actualiza oportunamente su memoria global (g_{best}) y se traslada para el siguiente punto de captura, que en este caso es x_2. Siguiendo el mismo principio explicado, el pescador continúa con su pesca hasta el último punto de captura x_3. Como ya hemos dicho este procedimiento se repite T iteraciones y entonces el algoritmo devuelve como resultado una función de *ranking* construida con el vector de posición relativo a la memoria global.

Durante todo el proceso de pesca, nuestro algoritmo implementa una estrategia de reinicios de puntos de captura para evitar zonas de estancamiento y convergencias tempranas hacia mínimos locales. Esta estrategia consiste en identificar y cambiar la posición de aquellos puntos de captura no prometedores para un rango de iteraciones DP (por ejemplo, 5 iteraciones).

Con este objetivo, durante cada iteración se registran aquellos puntos de pesca en los cuales el pescador no alcanzó a superar el mejor valor de aptitud logrado hasta el momento por cada punto de captura en particular. Cuando el desarrollo de la pesca en un determinado punto de captura sea cero durante la cantidad de iteraciones especificadas en DP, consideramos entonces que la pesca en este punto está estancada (presencia de convergencia temprana o punto de búsqueda muerto) y, por lo tanto, procedemos a definir una nueva posición para dicho punto de captura.

Este cambio de posición de los puntos de la captura no prometedores puede ejecutarse de dos maneras:

1. Inicializando aleatoriamente cada uno de estos puntos de captura.

2. Moviendo de una forma controlada la posición de los puntos de captura hacia zonas de pesca nuevas o no exploradas. En este caso, sería necesario conocer qué zonas de pesca han sido visitadas o no por el pescador.

Este proceso de chequear el progreso de la pesca en cada uno de los puntos de captura será repetido hasta la última ronda de iteraciones del método.

Finalmente, el modelo de *ranking* f es construido con el vector de posición (g_{best}) obtenido en la última iteración.

Todo este procedimiento general que lleva a cabo RankFSP puede ser analizado siguiendo la secuencia del Algoritmo 6.1.

Algoritmo 6.1: Algoritmo RankFSP

Input: $S = \{(q_i, d_i, y_i)\}_{i=1}^{m}$, E, T, N y M
for i = *1,...,*N **do**
  Inicializar aleatoriamente x_i;
  Evaluar x_i;
  Actualizar p_i;
  Actualizar g_{best};
end
for k = *1,...,*T **do**
  for i = *1,...,*N **do**
    while *(se encuentren mejoras)* **do**
      for j = *1,...,*M **do**
        Actualizar y_{ij} , $y_{ij} = x_i + A_j$;
        Evaluar y_{ij} mediante $R(f) = \sum_{i=1}^{m} (1 - E(\pi_i, y_i))$;
      end
      Actualizar x_i y p_i;
      Aplicar estrategia para actualizar coeficiente de amplitud, c;
    end
    Actualizar g_{best};
  end
  Aplicar estrategia de reinicios de puntos de captura;
end
Construir la función de ranking f con el vector de posición g_{best};
Result: f

Ventajas del algoritmo: Podemos concluir que RankFSP al igual que RankPSO es un método de L2R que puede ser clasificado dentro de los métodos que optimizan directamente cualquier medida de evaluación y que pertenece a la categoría de aproximación por listas (*list-wise approach*). Más específicamente, este puede ser ubicado en la sub-categoría de aquellos métodos que usan tecnologías especialmente diseñadas para optimizar medidas no suaves de RI.

Además de las similitudes generales que este grupo de métodos presentan en relación a las ventajas que brinda la optimización directa de medidas de evaluación. RankFSP posee ciertas diferencias beneficiosas sustentadas en su propia concepción. Así, usa un diseño

simple a través de una función de *ranking* lineal e implementa un comportamiento multi-arranque (reinicia la búsqueda a partir de nuevos puntos de captura) dirigido a evitar zonas de estancamiento y convergencias tempranas hacia mínimos locales.
Por otro lado, a pesar de que RankFSP y RankPSO sigan el mismo objetivo, se ubiquen en la misma categoría e implementen un modelo basado en una función de *ranking* lineal, también presentan entre sí diferencias bien marcadas. En este sentido, RankPSO toma las ventajas innatas de la inteligencia de enjambre y sus partículas establecen una comunicación de información que les permite una cooperación efectiva en busca de una buena solución al problema. Sin embargo, en algunas ocasiones su tendencia es a la convergencia temprana y no aprovechan las ventajas de la búsqueda local. En el caso de RankFSP sí contempla una buena estrategia para llevar a cabo la búsqueda local y, además, garantiza una buena exploración del espacio de búsqueda. Por otra parte, RankPSO espera que todos sus agentes estén estancados para aplicar una estrategia de diversificación, y RankFSP, va chequeando sus agentes de forma independiente, moviendo o reiniciando los que se queden estancados.

6.3. Preprocesamiento de las colecciones de datos

En este captulo utilizaremos las dos ltimas colecciones de datos propuestas por *Microsoft Research*, MSLR-WEB10K y MSLR-WEB30K, para llevar a cabo la tarea del L2R. En estas colecciones de gran tamaño, el *dataset* MSLR-WEB30K consta de más de 30.000 consultas, y MSLR-WEB10K abarca cerca de 10.000 consultas las cuales constituyen un subconjunto de MSLR-WEB30K. En estos datasets cada documento est representado por 136 rasgos extrados a partir de la consulta correspondiente.

6.3.1. Justificación

En el capítulo 4 de esta memoria, además de la propuesta y validación de RankPSO como método de L2R, nos enfocamos en cómo mejorar la disposición de información que tienen los conjuntos típicos de datos de la colección .Gov de LETOR 3.0, con el objetivo de garantizar que los métodos de L2R tuviesen un mejor rendimiento. Para tratar estos

datasets, que contaban con una enorme proporción de documentos irrelevantes con respecto aquellos etiquetados como relevantes, la propuesta de un procedimiento combinado de dos pasos (ver detalles en la Sección 4.3.2.) para reducir la cantidad de documentos irrelevantes a nivel de consulta resultó ser una decisión acertada.

Por tanto, cuando nos planteamos a priori que los conjuntos de datos: MSLR-WEB10K y MSLR-WEB30K (ver detalles en Sección 2.8) son aproximadamente 20 y 62 veces más grandes que los *datasets* estudiados (por ejemplo, HP2004 y TD2003), resulta evidente pensar que quizás este procedimiento sea también apropiado para eliminar algunas incongruencias (documentos ruidosos, mal etiquetados, etc.) que limitan una buena generalización del modelo de *ranking* y, por ende, un mejor rendimiento en la fase de prueba. Sin embargo, la estructura documental y la disposición de las etiquetas de relevancia en estos grandes *datasets* nos imponen un nuevo reto, debido a que en estos conjuntos tenemos un promedio de 120 documentos por consulta, existen cinco niveles de relevancia y en la mayoría de las consultas se tiene un balance entre la cantidad de documentos relevantes y los no relevantes. Por lo cual, la idea de llevar a cabo una reducción de instancias (documentos irrelevantes) a nivel de consulta ya no es oportuna.

Ante este situación, nos seguimos planteando que, a pesar de la utilidad evidente de tales colecciones, el gran tamaño de estos conjuntos de datos puede provocar inconvenientes en la tarea del L2R. Por ejemplo:

- Un aumento considerable del tiempo de respuesta de los modelos de L2R. Cuantas más consultas y documentos hayan, mayor será el tiempo necesario para construir y ajustar el modelo al mejor *ranking* en cada consulta.

- Se aumenta la sensibilidad al ruido y la posibilidad de sobreajuste (*overfitting*) de los modelos sobre el conjunto de entrenamiento. Al emplear un mayor número de datos es más probable que se retengan ejemplos ruidosos. Eso provocará que esos ejemplos de escasa calidad afecten a los métodos de L2R en su generalización del modelo y los lleve a un sobreajuste.

Por lo tanto, consideramos sugerente y apropiado, con la intención de disminuir el tiempo de aprendizaje (construcción) de los modelos de L2R, alcanzar un entrenamiento escalable

y eficiente, y potenciar mayores prestaciones en la etapa de prueba; reducir la dimensionalidad de los datos, manteniendo la información esencial y tratando de obtener una representación más compacta del conjunto de datos.

Nos dirigiremos específicamente a la reducción de datos como estrategia para el preprocesado de estas colecciones de datos, y como no pensamos que resulte conveniente la selección o eliminación de documentos, nos estaremos enfocando en la selección de rasgos.

En el siguiente apartado se mencionan los inconvenientes de utilizar los métodos de selección de rasgos diseñados para clasificación en problemas de *ranking*, se describen algunos métodos de selección de rasgos desarrollados para L2R y finalmente se propone un nuevo método de selección de rasgos encaminado a mejorar la tarea del *ranking* sobre los *datasets* que se estudian.

6.3.2. Selección de rasgos

Con el tiempo, el número de rasgos usados en el L2R ha aumentado drásticamente. Aunque el número creciente de rasgos propician más información a los algoritmos de *ranking*, este está relacionado directamente con la complejidad computacional y en cierta magnitud y para diversos casos al sobreajuste del modelo de *ranking* [124]. En muchos problemas de aprendizaje automático, cuando el número de rasgos seleccionados aumenta considerablemente, los rendimientos no mejoran, y en varios casos, estos siempre tienden a disminuir, lo cual valida el hecho que el uso de más rasgos no necesariamente nos conduce a superiores rendimientos en el *ranking*. Por ende, la selección de rasgos se convierte inevitablemente en un asunto importante y en un medio poderoso para evitar el sobreajuste de los modelos [125].

Aunque la selección de rasgos para el *ranking* sea vital, la mayoría de estos métodos que han sido usados para mejorar tareas de *ranking* fueron desarrollados inicialmente para clasificación.

Básicamente, cuando aplicamos estos métodos de selección de rasgos a la ordenación, pueden surgir varios problemas:

- Primero, existe una brecha significante entre clasificación y ordenación. En la ordenación, se utiliza un número de categorías ordenadas, representando una relación de *ranking* entre instancias; mientras que en clasificación las categorías no están ordenadas. Obviamente, los métodos de selección de rasgos para clasificación no son convenientes para la ordenación [124].

- Segundo, las medidas de evaluación de RI (por ejemplo, MAP y NDCG) usadas en problemas de ordenación son diferentes de aquellas medidas usadas en clasificación: (1) en la ordenación usualmente la precisión es más importante que la exhaustividad (*recall* en inglés) [126], mientras que en la clasificación, ambas, precisión y exhaustividad, son importantes; (2) en el *ranking*, ordenar correctamente las primeras n instancias resulta más crítico [44], mientras que en clasificación tomar una decisión sobre la clasificación correcta resulta de igual importancia para todas las instancias.

Ante esta problemática, sería prudente desarrollar métodos específicos de selección de rasgos para la ordenación. A pesar de esta necesidad, en los últimos años se han desarrollado pocas propuestas [124, 127–130] de selección de rasgos encaminadas al L2R. Geng et al. [124] proponen un método eficiente de selección de rasgos para el *ranking*, que utiliza un algoritmo de búsqueda voraz (*greedy*) enfocado en encontrar aquellos rasgos que simultáneamente tengan el valor máximo de importancia y el valor mínimo de semejanza, a nivel de todo el conjunto de datos. A diferencia, Dang y Croft [128] usan una estrategia de búsqueda primero el mejor para proponer un subconjunto de rasgos representativos y luego emplean el algoritmo de ascenso por coordenadas para aprender los pesos de dichos rasgos sobre cualquier medida de evaluación definida sobre un conjunto de entrenamiento. Este enfoque demostró superar el enfoque voraz mencionado anteriormente. Por otro lado, Pan et al. [127] presentan tres propuestas de selección de rasgos usando *boosted trees*, e incluyendo enfoques voraces y aleatorios. En todos los métodos, los *boosted trees* calculan el nivel de importancia para cada rasgo. En la primera propuesta, un enfoque voraz lleva a cabo la selección de aquellos rasgos con una alta relación de importancia. En un segundo algoritmo, se seleccionan aquellos rasgos más importantes, pero menos similares a otros rasgos. Y el tercer método, incorpora una selección de rasgos aleatoria, donde se realiza

una eliminación hacia atrás (*backward elimination*) basada en la importancia de los rasgos. Por otro lado, Gupta y Rosso [129] proponen un enfoque basado en divergencia esperada para seleccionar un subconjunto de rasgos altamente discriminativos sobre categorías de relevancia, donde la importancia de los rasgos son estimadas por el valor de las puntuaciones de evaluación producidas individualmente por cada uno de estos. Lai et al. [130] proponen una formulación de optimización convexa conjunta la cual minimiza errores de *ranking* mientras simultáneamente conduce la selección de rasgos.

A pesar de los resultados obtenidos, todavía estos métodos no logran una buena selección de rasgos, y otros presentan inconvenientes desde su formulación. Por ejemplo, los métodos de selección de rasgos propuestos en [124, 127] pueden manifestar un comportamiento sesgado y, por tanto, resultarle difícil decidir qué número de rasgos seleccionado es el apropiado. Por otro lado, en la fase de selección de rasgos (la primera fase) de algunos de los métodos existentes, seleccionar directamente los rasgos nos puede llevar a problemas de optimización binaria (por ejemplo, la expresión (2) en [124]), los cuales se conocen por ser no convexos y difíciles de analizar.

Todo lo cual deja abierto un campo muy prometedor y necesario para mejorar las tareas de *ranking* en particular.

Propuesta de selección de rasgos.

Específicamente, nuestra propuesta está encaminada a desarrollar un procedimiento simple y efectivo de selección de rasgos. Este procedimiento está basado en la idea de que existen rasgos que a nivel de colección parecen propiciar el mejor rendimiento para llevar a cabo la disposición del *ranking* pero que, en esencia, estos valores de precisión que alcanzan están sustentados y sesgados por sólo un grupo de consultas y no la totalidad del conjunto. Esto podría llevarnos a crear un modelo de *ranking* que va bien para algunas características de algunas consultas, pero que no contempla toda la información necesaria para realizar a plenitud una discriminación de relevancia sobre un nuevo conjunto de datos. Con esta premisa, sería conveniente considerar y llevar a cabo una selección de los rasgos más representativos a nivel de consulta, potenciando con este conjunto de rasgos una sinergia en cuanto a rendimiento a nivel de colección. Esto evitará también, que los métodos de

L2R en la etapa de prueba presenten bajos rendimientos por la pobre generalización de sus modelos de *ranking*. Como ventaja añadida, este procedimiento está diseñado para ejecutar el mínimo de evaluaciones de la función objetivo, que tanto para esta propuesta de selección de rasgos como para las mencionadas, siempre está relacionado con una medida de evaluación de RI.

Nuestro procedimiento de selección de rasgos consta de cuatro pasos simples:

1. Se calcula para cada una de las consultas, la importancia que ejerce en la ordenación cada uno de los rasgos de manera individual. El valor resultante de evaluar en una consulta el rendimiento en el *ranking* de cada rasgo, en términos de la precisión media (AP) será considerado como su puntuación o valor de importancia para dicha consulta. En este paso, después de calcular la importancia del primer rasgo, se etiquetan y descartan aquellas consultas cuyo valor AP sea igual a cero, posibilitando una disminución considerable del tiempo de respuesta del procedimiento.

2. Por cada consulta, se ordenan descendentemente los rasgos según las puntuaciones de importancia calculadas y se determinan cuáles son los mejores y peores rasgos, es decir, aquellos rasgos que según su información garantiza un mejor *ranking* (tienen asociado el mejor valor de AP) al corpus documental de la consulta y aquellos que peor lo hacen.

3. Basado en estos resultados por cada consulta, se calcula entonces y a nivel de conjunto de datos una ponderación positiva y una ponderación negativa por cada rasgo, donde la ponderación positiva de un rasgo específico se corresponderá con un valor entero que representa la cantidad de consultas donde dicho rasgo fue seleccionado como el mejor o fue incluido entre los mejores. Por su parte, el valor de ponderación negativo de un rasgo resultará del conteo de la cantidad de consultas donde dicho rasgo fue considerado como el peor o incluido entre los peores. Se resta entonces, la ponderación negativa a la ponderación positiva por cada rasgo, siendo utilizados estos valores resultantes para ordenar descendentemente todos los rasgos. Estos valores obtenidos pueden ser interpretados como la tendencia de dichos rasgos a contribuir a un mejor o peor rendimiento en la tarea del *ranking*. Cuanto mayor sea un valor

resultante positivo indica que el rasgo ejerce una mayor influencia en el *ranking* para esa misma cantidad de consultas; y de manera contraria se manifiesta para el caso de valores negativos.

4. Finalmente, se seleccionan de la lista ordenada los primeros rasgos que en su conjunto cubran positivamente un umbral δ (%) del total de consultas del conjunto de datos. Este valor de δ es calculado descartando aquellas consultas cuyo valor AP es igual a cero. Se validó empíricamente, haciendo uso del conjunto de validación que el subconjunto de rasgos que cubren positivamente un δ igual al 60 % del total de consultas, muestra en la mayoría de los casos, igual o mejor rendimiento que otros subconjuntos de mayor tamaño, con lo cual, y buscando una mejor representatividad y compactación de la información, nos indica que este porcentaje puede ser considerado como referencia para decidir qué rasgos serán seleccionados.

Para ilustrar este procedimiento con un ejemplo, tomemos el conjunto de entrenamiento del Fold1 del *dataset* MSLR-WEB10K. En este fichero de datos, cada fila corresponde a un par consulta-documento. Este par está representado por un vector de rasgos de 136 dimensiones. Los detalles de cada uno de estos rasgos pueden ser consultados en el sitio de *Microsoft Research*, específicamente en el proyecto de *Microsoft Learning to Rank Datasets*[1].

Iniciando el procedimiento, ordenamos las instancias de la primera consulta usando los valores del rasgo con ID n^o1. A partir de este ranking obtenido, evaluamos el rendimiento en términos del criterio de evaluación AP y tomamos este valor resultante como la puntuación de importancia para este rasgo. De igual manera, calculamos la importancia de este rasgo para cada una de las 6.000 consultas del conjunto de entrenamiento. Durante este proceso, se identifican un total de 559 consultas cuyo valor de AP es cero. Estas consultas que no aportan información alguna son etiquetadas de tal forma que cuando se repite el procedimiento para calcular las importancias de los restantes rasgos son excluidas del procesamiento. Esta simple consideración repercute en que nos ahorremos evaluar un total de 75.465 consultas para obtener el mismo resultado final.

[1]Disponible en http://research.microsoft.com/en-us/projects/mslr/feature.aspx

Luego, enfocados en la primera consulta, ordenamos descendentemente todos los rasgos acorde a sus puntuaciones y seleccionamos el mejor o mejores rasgos (mayor valor de AP), así como el peor o peores rasgos (menor valor de AP). Este procedimiento es repetido para cada una de las restantes consultas.

Basado en estas selecciones por cada consulta, vamos ponderando positiva y negativamente a cada uno de los rasgos. Por ejemplo, el rasgo n^o1 garantiza el mejor *ranking* en 40 consultas y el peor rendimiento en 54 consultas, en las demás consultas manifiesta un comportamiento promedio. Por tanto, el rasgo n^o1 tiene una ponderación final negativa cuyo valor es -14. De igual manera, calculamos el peso final de los restantes rasgos. Estos valores de ponderación obtenidos pueden ser interpretados como la tendencia de dichos rasgos a contribuir a un mejor o peor rendimiento en la tarea del *ranking* para tal indicada cantidad de consultas.

A partir de estas ponderaciones finales ordenamos descendentemente todos los rasgos, y comenzando por el primer rasgo, el segundo y así de forma consecutiva, vamos conformando un subconjunto de rasgos que en su totalidad tienen que cumplir con la condición de influir positivamente en el 60 % del total de consultas del conjunto de entrenamiento. Para este caso de estudio en específico, son los 30 primeros rasgos los que cumplen con esta condición y, por tanto, serán considerados como el subconjunto representativo de rasgos de este conjunto de entrenamiento.

6.4. Experimentación

En esta sección, presentamos los resultados experimentales de la fase de evaluación de nuestro método propuesto, RankFSP, en términos de su efectividad frente a los conjuntos de datos MSLR-WEB10K y MSLR-WEB30K. En este análisis seguimos los esquemas experimentales definidos en LETOR para poder realizar comparaciones directas con los principales métodos que establecen el estado del arte en L2R. Como medidas de evaluación, usamos *Mean Average Precision* (MAP), *Normalized Discounted Cumulative Gain* (NDCG) y la *precision at n* (P@n), considerando en estas dos últimas las 10 primeras posiciones del *ranking*. Analizamos además, las potencialidades del procedimiento propuesto

de selección de rasgos, para mejorar el rendimiento de los métodos de L2R durante la tarea del *ranking*. Los resultados obtenidos y disponibles fueron analizados estadísticamente. Finalmente, se realizó un análisis evaluativo del coste computacional de RankFSP y el procedimiento de selección de rasgos, considerando la cantidad de consultas evaluadas y el tiempo de procesamiento.

6.4.1. Evaluación de rendimientos obtenidos y comparación de métodos

Para poner en acción nuestro método propuesto, establecimos los siguientes parámetros claves: 10 iteraciones, 25 puntos de captura y una red de pesca compuesta por 10 vectores de posición. Los lanzamientos de la red de pesca en cada punto de captura se realizarán continuamente mientras se encuentren mejores soluciones para dicho punto. En caso contrario, el pescador (algoritmo) pasa al siguiente punto de captura. Cómo en nuestro método la cantidad de lanzamientos está controlada por un criterio de parada y no prefijada a un valor fijo, la cantidad de evaluaciones totales a realizar oscilan en el rango de 2.500 a 5.000 evaluaciones para las colecciones utilizadas. La medida de evaluación a aplicar fue MAP y el coeficiente de amplitud c fue multiplicado continuamente por un factor de valor 0,95 cada vez que lanzamos más de una vez la red de pesca en un determinado punto de captura. El reinicio de los puntos de pesca no mejorados fue ejecutado para un DP de valor 5, la redefinición o reubicación de estos puntos se realizó de forma aleatoria, siguiendo una distribución uniforme. La mayoría de los valores que asignamos a estos parámetros fueron ajustados a través de pruebas empíricas.

Una vez definidos los parámetros, y con el objetivo de evaluar el comportamiento de RankFSP ante otras propuestas referenciadas en la literatura, nosotros seguimos las especificaciones experimentales publicadas en el sitio web de LETOR, que están en correspondencia total con las propuestas por *Microsoft Research* en su sitio web[2] para evaluar sus dos grandes *datasets*: MSLR-WEB10K y MSLR-WEB30K.

Se llevó a cabo en todas las pruebas realizadas una validación cruzada de 5-*fold*. Cada *fold* incluye todas las consultas del *dataset* correspondiente y consta de tres ficheros: el conjunto de entrenamiento, el de validación y el de prueba. El primero incluye el 60 % de

[2]Disponible en URL: http://research.microsoft.com/en-us/projects/mslr/

las consultas, el segundo un 20 % y las restantes consultas (también un 20 %) pertenecen al conjunto de prueba.

Los conjuntos de entrenamiento fueron transformados, es decir, reducidos siguiendo el procedimiento de selección de rasgos explicado detalladamente en la Sección 6.3.2. Además de este proceso de reducción de datos, fue aplicado también un filtrado a nivel de consulta, con el objetivo de eliminar todas aquellas consultas que no aportasen información a los modelos de L2R a la hora de llevar a cabo la construcción de funciones de *ranking*. Este procedimiento de filtrado consta de dos pasos simples:

1. Eliminar todas aquellas consultas que por su corpus documental irrelevante no aportan nada al modelo de aprendizaje. Entiéndase las consultas que no poseen ningún documento etiquetado como relevante. En tales casos, el valor resultante de aplicar cualquier medida de evaluación siempre será cero.

2. Eliminar las consultas consideradas como *outliers*[3], es decir, aquellas consultas que según su entorno (conjunto de datos al que están inscritas) tienen exageradamente el mayor número de documentos relevantes. En este caso, realizamos un análisis estadístico exploratorio[4] condicionado al porcentaje de documentos relevantes con respecto al total de documentos recuperados por cada consulta, pudiendo determinar y delimitar cuales son los *outliers*. Este último paso puede evitar además, entrenar modelos sesgados hacia aquellas consultas con mayor número de documentos relevantes.

La aplicación de este filtrado eliminó de cada conjunto de entrenamiento de MSLR-WEB10K un promedio de 12,4 % de las consultas, y un 12,2 % en los conjuntos de MSLR-WEB30K.

Posterior a estas etapas de preprocesamiento, el algoritmo RankFSP ejecutó su proceso de entrenamiento sobre dichos conjuntos reducidos para cada *fold*. Una vez construido el modelo de *ranking* fue entonces validado y probado considerando las medidas de evaluación

[3]Un valor atípico (en inglés outlier) es una observación que es numéricamente distante del resto de los datos. Estos valores extremos pueden sesgar los coeficientes de correlación y las líneas de mejor ajuste en la dirección equivocada.

[4]Para explorar los datos e identificar estos valores extremos fue usado el gráfico de tallo y hojas, y el diagrama de caja, generados por el software de análisis estadístico SPSS (disponible en http://www.ibm.com/software/analytics/spss/).

correspondientes. Según las pautas experimentales de LETOR, los cinco resultados obtenidos (uno por cada *fold*) de la fase de prueba son promediados y, finalmente, este valor resultante puede ser comparado directamente en cuanto a precisión con otros algoritmos publicados.

Para tener un indicador confiable de las ventajas para el L2R que brinda el método propuesto de selección de rasgos sobre estos *datasets*, en un primer momento aplicaremos RankFSP directamente sobre los conjuntos de entrenamiento sin reducir y etiquetaremos el método como *RankFSP* y, para el caso donde apliquemos RankFSP sobre los conjuntos reducidos, utilizaremos la etiqueta *RankFSP+Prep*. Para hacer más interesante y amplia la comparación, incorporamos también al método RankPSO, bajo las mismas condiciones y usando intuitivamente el mismo modo de etiquetado.

Además de estos dos métodos, vamos a implementar e incorporar en este trabajo otra propuesta de modelo de L2R basada en una heurística -reciente y poco explorada- de optimización global denominada Supernova [131]. Haciendo una breve síntesis del fenómeno físico, Supernova sería una explosión de partículas que se organizan en el tiempo de acuerdo a su velocidad y masa siguiendo las leyes de la conservación de la energía. En esencia es un proceso que desde su inicio caótico llega a un equilibrio mas ordenado, es decir, mejora su orden. La explosión inicial se puede emular como aleatoriedad y la interacción de partículas expulsadas restringirse a dos interacciones: una que expande y otra que atrae las partículas. Aunque es un fenómeno sumamente complejo, por simplicidad, este algoritmo sólo considera dos de las fuerzas presentes en el mismo: el impulso y la gravitación. En este sentido, una heurística que emule las etapas del fenómeno de las supernovas podría encontrar regiones importantes dentro de una región de búsqueda pues las soluciones se aglomerarían alrededor de mejores soluciones por la atracción que ejercen estas sobre otras no tan buenas, tal como sucede con las partículas expulsadas de las supernovas físicas. Esta heurística sigue cuatro etapas básicas simulando el fenómeno físico, la explosión (donde se generan k soluciones aleatorias), la evaluación (de cada partícula según la función objetivo), los cálculos (de vectores de impulso y los correspondientes a la atracción gravitatoria) y la reinicialización (donde se refina la búsqueda a partir de la mejor partícula encontrada). En nuestro proceso de adaptación y aplicación de Supernova

al L2R consideramos que cada partícula es una solución, es decir, un vector de pesos para construir la función de *ranking* sobre el conjunto de entrenamiento. El tamaño del vector de cada partícula generada se corresponderá con el número de rasgos de la colección. La masa de cada partícula se corresponderá con el valor que devuelva la función objetivo definida directamente por la medida de evaluación MAP. El impulso estará representado como el movimiento de la partícula en su trayectoria original (donde la velocidad es un componente aleatorio), y el potencial gravitatorio es el movimiento de la partícula hacia la mejor solución.

Siguiendo las recomendaciones de la etapa de parametrización en Supernova, para esta fase de experimentación los parámetros fueron fijados de la siguiente manera: 100 iteraciones, 25 partículas, la constante que representa el peso que tiene el impulso, es decir, la energía potencial gravitatoria F igual a 1, la constante que representa el peso que tiene la fuerza de gravitación universal G también con valor 1, como fuerza de explosión λ será considerado el valor promedio del límite máximo y mínimo del espacio de búsqueda, la tasa de disminución en el tiempo de la fuerza de explosión se corresponderá con un α de 1,1 y serán ejecutados 10 reinicios. A este nuevo método lo etiquetaremos como *RankSupernova*, y cuando sea aplicado sobre los conjuntos reducidos le referiremos como *RankSupernova+Prep*.

Por otro lado, a diferencia de lo que ocurre con otras colecciones comúnmente empleadas como OHSUMED, Gov y Gov2, para las cuales sus resultados experimentales con los principales métodos de referencia dentro del L2R son publicados en LETOR, para el caso de MSLR-WEB10K y MSLR-WEB30K no ocurre así, es decir, todavía no se ha publicado ningún resultado en esta web.

Por tanto, hemos utilizado los resultados experimentales de artículos [132] [133] publicados hasta la fecha, que hayan realizado pruebas y evaluaciones de métodos frente a estos dos *datasets*. En este sentido, como los trabajos son relativamente pocos y no siempre los resultados publicados abarcan todas las medidas de evaluación consideradas en este apartado, no nos hemos enmarcado en aquellos métodos de L2R del mismo tipo o categoría que nuestro modelo, simplemente nos hemos comparado con todos los métodos que hemos encontrado. Tenemos métodos que utilizan en la concepción de su modelo una función

de *ranking* lineal (por ejemplo, AdaRank-MAP [9], RankingSVM(RSVM)[5] [16]), otros emplean una función de *ranking* no lineal (por ejemplo, RankBoost [17]) y otros llevan a cabo la tarea del L2R en línea (*online*) como PARank-NDCG [133], Committee Perceptron (ComP) [134], para este caso implementado en C++ por Suhara en [133], y Stochastic Pairwise Descent (SPD) [135] implementado con *sofia-ml*[6]. Para este último algoritmo, se emplearon dos versiones utilizando Pegasos-SVM (SPD-SVM) y Passive-Aggressive (SPD-PA) cómo métodos de aprendizaje ya incorporados en el paquete de *sofia-ml*.

La Tabla 6-2 muestra la precisión en el *ranking* lograda por cada uno de los métodos sobre los *datasets*: MSLR-WEB10K y MSLR-WEB30K, considerando MAP como medida de evaluación. Los resultados en MSLR-WEB10K muestran como RankFSP alcanza un rendimiento muy competitivo con respecto a las demás propuestas algorítmicas de L2R. Además, se observa como la incorporación del procedimiento de selección de rasgos como etapa previa al entrenamiento eleva las prestaciones de los métodos RankFSP, RankPSO y RankSupernova, obteniendo *RankFSP+Prep* el mejor resultado en el *ranking*. Este comportamiento se manifiesta de igual forma en los resultados obtenidos por RankFSP, RankPSO y RankSupernova frente a MSLR-WEB30K, cuyos valores pueden ser consultados en la parte (b) de dicha tabla.

Entretanto, la Tabla 6-3 muestra el comportamiento de los valores de precisión en el *ranking* obtenidos por cada método bajo los términos de las medidas NDCG@1 a NDCG@10. En este caso, el rendimiento de nuestro método propuesto *RankFSP+Prep* reafirma su condición de mejor método en las primeras posiciones del *ranking*. Se observa además, como los valores de *RankPSO+Prep* le siguen muy de cerca, y que estos dos métodos junto a *RankSupernova+Prep* establecen con respecto a la medida NDCG una brecha significante con relación a los resultados alcanzados por cada uno de los demás métodos de L2R evaluados.

[5]Usando svm_rank como implementación disponible en URL: http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

[6]Acrónimo de *Suite of Fast Incremental Algorithms for Machine Learning*, disponible en URL: https://code.google.com/p/sofia-ml/

[7]Para obtener estos resultados, Balasubramanian en [132] usa el paquete R Lasso2 disponible en http://cran.r-project.org/web/packages/lasso2/index.html

(a) MSLR-WEB10K

Algoritmo	**MAP**	**Breve descripción**
BM25	0,2561	*Single feature ranker* [42]
RankSupernova	0,2739	Algoritmo de optimización basado en Supernova [131]
RankBoost	0,2770	*Boosting algorithm* que minimiza la discordancia a pares en el *ranking* [17]
AdaRank	0,2840	*Boosting algorithm* que optimiza directamente a MAP [9]
RankPSO	0,2948	Algoritmo de optimización directa inspirado en PSO [100]
L1-LogReg	0,3031	Logistic Regression with L1-constraints[7]
RankFSP	0,3044	Algoritmo de optimización directa basado en FSP [119]
RankSupernova+ Prep	0,3048	Procedimiento combinado de selección de rasgos, filtrado y RankSupernova
AFS	0,3053	Co-ordinate ascent-based algorithm with direct optimization for MAP [136]
RankPSO+Prep	0,3193	Procedimiento combinado de selección de rasgos, filtrado y RankPSO
RankFSP+Prep	0,3306	Procedimiento combinado de selección de rasgos, filtrado y RankFSP

(b) MSLR-WEB30K

Algoritmo	**MAP**	**Breve descripción**
RankSupernova	0,2785	Algoritmo de optimización basado en Supernova [131]
RankPSO	0,2952	Algoritmo de optimización directa inspirado en PSO [100]
RankFSP	0,3021	Algoritmo de optimización directa basado en FSP [119]
RankSupernova+ Prep	0,3062	Procedimiento combinado de selección de rasgos, filtrado y RankSupernova
RankPSO+Prep	0,3185	Procedimiento combinado de selección de rasgos, filtrado y RankPSO
RankFSP+Prep	0,3312	Procedimiento combinado de selección de rasgos, filtrado y RankFSP

Tabla 6-2: Rendimientos alcanzados por algoritmos de L2R sobre los *datasets* MSLR-WEB10K (a) y MSLR-WEB30K (b), considerando MAP como medida de evaluación.

Es notorio mencionar además que nuestros métodos *RankFSP+Prep* y *RankPSO+Prep* alcanzan tales resultados con un valor máximo de 5.000 evaluaciones completas. Según [133], el método PARank-NDCG empleó 60.000 iteraciones para procesar MSLR-WEB10K y 180.000 frente a MSLR-WEB30K; y el número de iteraciones en los algoritmos ComP, SPD-SVM y SPD-PA fue fijado a 100.000 para MSLR-WEB10K y en 300.000 para MSLR-WEB30K.

Por otra parte, en la Tabla 6-4 sólo incluimos los resultados de precisión de RankFSP, RankPSO y RankSupernova, cada uno con sus dos variantes de presentación, es decir, con y sin preprocesamiento de los conjuntos de entrenamiento. El resto de los métodos de

(a) MSLR-WEB10K

	NDCG									
Algoritmo	**@1**	**@2**	**@3**	**@4**	**@5**	**@6**	**@7**	**@8**	**@9**	**@10**
PARank-NDCG	0,3182	0,3211	0,3276	0,3330	0,3380	0,3423	0,3471	0,3503	0,3541	0,3572
SPD-SVM	0,2871	0,2947	0,3039	0,3104	0,3161	0,3216	0,3267	0,3310	0,3350	0,3384
SPD-PA	0,2830	0,2987	0,3061	0,3131	0,3193	0,3246	0,3297	0,3338	0,3379	0,3415
ComP	0,2409	0,2515	0,2620	0,2697	0,2760	0,2820	0,2876	0,2923	0,2969	0,3008
RankingSVM	0,2990	0,3118	0,3214	0,3281	0,3339	0,3390	0,3442	0,3483	0,3527	0,3564
RankSupernova+Prep	0,3592	0,3530	0,3485	0,3446	0,3387	0,3332	0,3270	0,3208	0,3101	0,2979
RankPSO+Prep	0,4187	0,4128	0,4075	0,4018	0,3963	0,3844	0,3792	0,3701	0,3623	0,3574
RankFSP+Prep	0,4298	0,4237	0,4189	0,4112	0,4045	0,3952	0,3887	0,3799	0,3704	0,3630

(b) MSLR-WEB30K

	NDCG									
Algoritmo	**@1**	**@2**	**@3**	**@4**	**@5**	**@6**	**@7**	**@8**	**@9**	**@10**
PARank-NDCG	0,3208	0,3247	0,3304	0,3357	0,3408	0,3448	0,3490	0,3529	0,3567	0,3598
SPD-SVM	0,2845	0,2929	0,2997	0,3061	0,3123	0,3173	0,3218	0,3262	0,3304	0,3339
SPD-PA	0,2869	0,2997	0,3075	0,3147	0,3283	0,3344	0,3397	0,3445	0,3491	0,3533
ComP	0,2461	0,2585	0,2674	0,2752	0,2814	0,2869	0,2922	0,2971	0,3014	0,3056
RankingSVM	0,3008	0,3118	0,3212	0,3283	0,3344	0,3397	0,3445	0,3491	0,3533	0,3571
RankSupernova+Prep	0,3597	0,3540	0,3492	0,3450	0,3388	0,3335	0,3275	0,3212	0,3118	0,2984
RankPSO+Prep	0,4169	0,4115	0,4060	0,4007	0,3945	0,3829	0,3780	0,3710	0,3640	0,3550
RankFSP+Prep	0,4295	0,4229	0,4192	0,4123	0,4053	0,3944	0,3868	0,3771	0,3700	0,3607

Tabla 6-3: Rendimientos alcanzados en MSLR-WEB10K (a) y MSLR-WEB30K (b), considerando NDCG en las 10 primeras posiciones del *ranking*.

L2R: PARank-NDCG, SPD-SVM, SPD-PA, ComP, RankingSVM, RankBoost, AdaRank, L1-LogReg y AFS, no son incluidos en esta comparación debido a que no se tienen (no están publicados) los resultados de precisión (*Precision at n*) que alcanzan en ninguna de las posiciones del *ranking* frente a estos dos *datasets*.

(a) MSLR-WEB10K

Algoritmo	**P@1**	**P@2**	**P@3**	**P@4**	**P@5**	**P@6**	**P@7**	**P@8**	**P@9**	**P@10**
RankSupernova	0,3671	0,1902	0,1255	0,0813	0,0718	0,0611	0,0598	0,0588	0,0574	0,0527
RankPSO	0,4030	0,2132	0,1347	0,0890	0,0858	0,0642	0,0620	0,0602	0,0583	0,0523
RankSupernova+Prep	0,4470	0,2485	0,1472	0,0982	0,0924	0,0681	0,0655	0,0637	0,0622	0,0541
RankFSP	0,4592	0,2406	0,1493	0,0964	0,0911	0,0679	0,0657	0,0636	0,0618	0,0553
RankPSO+Prep	0,4840	0,2345	0,1277	0,0736	0,0709	0,0686	0,0659	0,0640	0,0624	0,0511
RankFSP+Prep	0,5197	0,2502	0,1409	0,0981	0,0912	0,0701	0,0655	0,0639	0,0622	0,0515

(b) MSLR-WEB30K

Algoritmo	**P@1**	**P@2**	**P@3**	**P@4**	**P@5**	**P@6**	**P@7**	**P@8**	**P@9**	**P@10**
RankSupernova	0,3669	0,1985	0,1243	0,0809	0,0713	0,0622	0,0596	0,0583	0,0569	0,0520
RankPSO	0,3908	0,1991	0,1110	0,0714	0,0678	0,0623	0,0590	0,0527	0,0481	0,0414
RankFSP	0,4425	0,2537	0,1741	0,1114	0,1059	0,0757	0,0691	0,0622	0,0589	0,0518
RankSupernova+Prep	0,4459	0,2511	0,1429	0,1098	0,0933	0,0699	0,0650	0,0631	0,0593	0,0530
RankPSO+Prep	0,4785	0,2214	0,1188	0,0802	0,0773	0,0701	0,0661	0,0633	0,0599	0,0540
RankFSP+Prep	0,5201	0,2514	0,1410	0,1078	0,0927	0,0792	0,0711	0,0664	0,0608	0,0582

Tabla 6-4: Rendimientos alcanzados en MSLR-WEB10K (a) y MSLR-WEB30K (b), considerando la medida *Precision at n* en las 10 primeras posiciones del *ranking*.

Finalmente, todos los resultados obtenidos (para cada *dataset* y medida de evaluación) no sólo validan los buenos rendimientos de RankFSP a la hora de acometer la tarea del *ranking*, sino también, lo eficiente que resultan los métodos propuestos de reducción de datos para propiciar mejoras en el rendimiento de métodos de L2R frente a estas colecciones.

6.4.2. Análisis estadístico

Después de haber obtenido y comparado los resultados de rendimiento de nuestro principal método propuesto RankFSP y de evaluar el comportamiento de un RankSupernova simple, se realizaron pruebas estadísticas para fundamentar con un basamento matemático el nivel de significación y comportamiento reflejados por la precisión alcanzada en la ordenación a nivel de consulta.

En este análisis sólo fueron considerados los métodos RankFSP, RankPSO y RankSupernova, incluyendo sus aplicaciones posteriores a la selección de rasgos y el filtrado, debido a que no están disponibles (publicadas) las precisiones a nivel de consulta de los demás algoritmos de L2R comparados en la sección anterior.

Específicamente fueron considerados los resultados de la precisión media (AP) obtenidas por cada algoritmo en función de cada una de las consultas pertenecientes a las colecciones de datos estudiadas. Para el caso de MSLR-WEB10K tenemos 10.000 valores de AP por cada algoritmo, y para MSLR-WEB30K un total de 31.531.

Se aplicaron entonces pruebas no paramétricas utilizando un modelo estadístico de comparación de medias para k muestras relacionadas o pareadas. Este modelo fue aplicado porque se ajusta bien a la necesidad de comparar diferentes algoritmos en un mismo grupo de consultas; todo lo cual permitió llevar a cabo un estudio comparativo más preciso de los rendimientos de cada algoritmo.

Específicamente, se realizó un análisis de varianza (ANOVA) [113] de 2 vías de Friedman por rangos [115] [137], incluyendo las comparaciones múltiples por parejas, y usando un intervalo de confianza del 95 %.

En la Figura 6-3 se muestran como las significancias asintóticas en ambas pruebas son

menores que 0,05 (0,00 en tales casos) y por tanto, rechazamos la hipótesis nula de igualdad de distribuciones y concluimos que existen por cada dataset, diferencias significativas entre al menos dos de los algoritmos implicados.

N	10.000
Chi-cuadrado	45.611,964
Grados de libertad	5
Sig. asintótica (prueba de dos caras)	,000

(a) MSLR-WEB10K

N	31.531
Chi-cuadrado	143.017,35
Grados de libertad	5
Sig. asintótica (prueba de dos caras)	,000

(b) MSLR-WEB30K

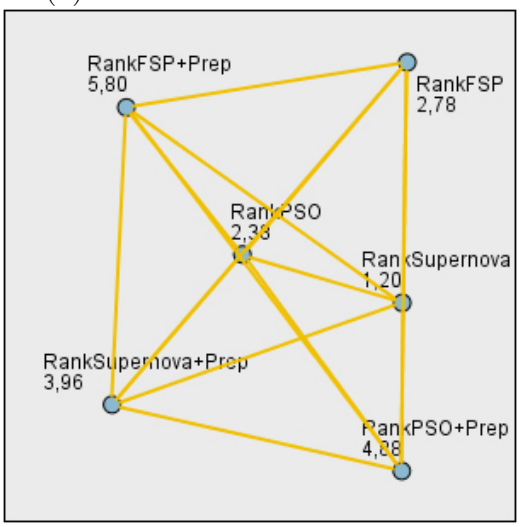
Figura 6-3: Estadísticos de contraste obtenidos para ambas colecciones de datos.

Entretanto, para el caso del *dataset* MSLR-WEB10K la Figura 6-4 visualiza como se desarrollaron las comparaciones por parejas a través de un gráfico de distancias de red (parte *a*) y una tabla de comparaciones (parte *b*). En el gráfico las distancias entre nodos de la red corresponden a las diferencias entre las muestras, y como estas son de color amarillas[8] podemos decir que existen diferencias estadísticas importantes entre todos los algoritmos. El valor de las etiquetas en los nodos se corresponden con el rango promedio calculado para cada algoritmo. En la parte (b), cada fila corresponde con una comparación de pareja distinta, donde en la columna "Prueba estadística" se detalla numéricamente cuán acentuadas están las diferencias significativas entre los algoritmos que se comparan. En este sentido, podemos percibir cómo para todos los casos comparados, los métodos *RankFSP+Prep*, *RankPSO+Prep* y *RankSupernova+Prep* mejoran con un alto nivel de significación estadística sus propuestas originales, es decir, a RankFSP, RankPSO y RankSupernova respectivamente, lo que reafirma los beneficios del método propuesto de selección de rasgos. Además, se observa como *RankFSP+Prep* supera significativamente a *RankPSO+Prep* y *RankSupernova+Prep*, y del mismo modo RankFSP refleja sus diferencias con relación a RankPSO y RankSupernova.

Para el caso del dataset MSLR-WEB30K las diferencias estadísticas entre los algoritmos

[8]En estos gráficos las líneas de color amarillo entre nodos representan diferencias significativas, mientras las de color negro indican que las diferencias no son notables.

(a) Gráfico de distancias de red



(b) Tabla de comparaciones

Muestra1-Muestra2	Prueba estadística	Error típico	Desv. Prueba estadística	Sig.	Sig. ady.
RankSupernova-RankPSO	-1,178	,026	-44,522	,000	,000
RankSupernova-RankFSP	-1,578	,026	-59,660	,000	,000
RankSupernova-RankSupernova+Prep	-2,756	,026	-104,182	,000	,000
RankSupernova-RankPSO+Prep	-3,675	,026	-138,910	,000	,000
RankSupernova-RankFSP+Prep	-4,594	,026	-173,637	,000	,000
RankPSO-RankFSP	-,400	,026	-15,137	,000	,000
RankPSO-RankSupernova+Prep	-1,578	,026	-59,660	,000	,000
RankPSO-RankPSO+Prep	-2,497	,026	-94,387	,000	,000
RankPSO-RankFSP+Prep	-3,416	,026	-129,115	,000	,000
RankFSP-RankSupernova+Prep	-1,178	,026	-44,522	,000	,000
RankFSP-RankPSO+Prep	-2,097	,026	-79,250	,000	,000
RankFSP-RankFSP+Prep	-3,016	,026	-113,977	,000	,000
RankSupernova+Prep-RankPSO+Prep	-,919	,026	34,727	,000	,000
RankSupernova+Prep-RankFSP+Prep	-1,838	,026	69,455	,000	,000
RankPSO+Prep-RankFSP+Prep	-,919	,026	-34,727	,000	,000

Figura 6-4: Resumen de las comparaciones múltiples por parejas de los algoritmos en MSLR-WEB10K.

se dan de igual forma que en MSLR-WEB10K. Para más detalle sobre los resultados en MSLR-WEB30K, consultar la figura del Apéndice D.

6.4.3. Análisis del coste computacional

En esta sección analizaremos de manera combinada el coste computacional de RankFSP y de la aplicación del procedimiento propuesto de selección de rasgos. Específicamente, nos enfocaremos en dos indicadores: tiempo de procesamiento y cantidad de consultas evaluadas.

Para ilustrar el comportamiento de estos dos indicadores utilizamos los ficheros de entrenamiento pertenecientes al Fold1 de cada conjunto de datos, es decir, de MSLR-WEB10K y MSLR-WEB30K. Como declaramos en la Sección 6.4.1, se establecieron como parámetros principales del método propuesto 10 iteraciones, 25 puntos de captura y una red de pesca compuesta por 10 vectores de posición, para una cota máxima de 5.000 evaluaciones considerando el comportamiento empírico de los lanzamientos de la red de pesca para estas colecciones.

El experimento fue desarrollado sobre una arquitectura Intel(R) Core(TM) i5 (4 CPUs) 2,53GHz y 8,0 GBytes de RAM.

En un primer momento, analicemos cómo varía el costo de tiempo durante el procesamiento de los conjuntos de entrenamiento, en dependencia de si incluimos o no, el procedimiento propuesto de selección de rasgos antes de la etapa de aprendizaje. Además de este procedimiento de reducción de datos, vamos a tomar en cuenta en todos los demás casos que serán analizados, el proceso de filtrado descrito en la fase de experimentación.

En la Tabla 6-5 podemos observar en resumen, el tiempo en horas que emplea el método de selección de rasgos y el proceso de filtrado al ser aplicado sobre cada conjunto de datos. Por ejemplo, el valor 5,20 representa la sumatoria de tiempo en horas que emplea el método de selección de rasgos para aplicar sus cuatro pasos sobre el conjunto de entrenamiento de MSLR-WEB10K. Es importante mencionar además, que aunque los tiempos parezcan un poco elevados, estamos tratando con las más grandes colecciones de datos para L2R y que particularmente el método de selección de rasgos incorpora además en cada uno de

sus pasos estrategias encaminadas a mejorar la tarea del *ranking*.

Conjunto de datos	**Selección de Rasgos**	**Filtrado a nivel de consultas**	**Total**
Train1 MSLR-WEB10K	5,20	0,04	5,24
Train1 MSLR-WEB30K	16,07	0,10	16,17

Tabla 6-5: Tiempo en horas empleado para el preprocesamiento de cada conjunto de entrenamiento.

Conjunto de datos	**RankFSP**	**Preprocesamiento + RankFSP**	**Reducción de tiempo (%)**
Train1 MSLR-WEB10K	189,28	64,72	65,81
Train1 MSLR-WEB30K	578,65	164,63	71,55

Tabla 6-6: Tiempo de procesamiento (horas) para construir una función de *ranking* a partir de cada conjunto de entrenamiento.

Siguiendo está lógica, en la Tabla 6-6 se presentan los tiempos resultantes de aplicar directamente sobre tales conjuntos de entrenamiento de MSLR-WEB10K y MSLR-WEB30K, nuestro método propuesto RankFSP y como segunda opción, combinándolo con la etapa de preprocesamiento propuesta. Los tiempos que se enmarcan en la columna encabezada como "Preprocesamiento + RankFSP", son la sumatoria de los valores de la columna "Total" de la Tabla 6-5 y el coste de tiempo que empleó RankFSP en construir su modelo sobre cada conjunto reducido de datos (fichero reducido de entrenamiento) en particular. Resulta también notorio el porcentaje de ahorro de tiempo de procesamiento que conlleva la aplicación del método de selección de rasgos y el filtrado sobre las colecciones estudiadas. Sería oportuno entonces, verificar las implicaciones de tales procedimientos en cuanto a la cantidad de consultas a evaluar.

Comenzando con la selección de rasgos, evaluamos para el rasgo n^o1 la totalidad de las consultas, es decir, 6.000, donde se determinó según el valor de la medida AP que 559 consultas no aportan información alguna para construir el modelo de *ranking*. Por tanto, el calcular la influencia de los restantes 135 rasgos, nos lleva a evaluar un total de 734.535 consultas (valor resultante de la expresión (6000-559)*135). Cómo con dicha información y otra adicional extraída durante la iteración inicial resulta suficiente para completar cada

uno de los pasos de la selección de rasgos, la totalidad de consultas finalmente evaluadas y analizadas asciende a 740.535.

Ahora, sobre este conjunto de entrenamiento reducido, aplicamos el proceso de filtrado que elimina para este caso 742 consultas. Aplicamos entonces nuestro método RankFSP, y tendremos que serán evaluadas un total de 26.290.000 consultas[9].

Siguiendo este esquema, una vez concluidas las etapas de preprocesamiento y entrenamiento, habrán sido evaluadas un total de 27.030.535 consultas.

Si por el contrario nos hubiésemos saltado la etapa de preprocesamiento, utilizando directamente RankFSP bajo las mismas condiciones, este hubiese tenido que evaluar un total de 30.000.000 consultas.

Este mismo análisis se realizó para MSLR-WEB30K y un resumen de ambos resultados se muestra en la Tabla 6-7, donde se especifica además, la cantidad de consultas que nos evitamos evaluar utilizando los procedimientos de selección de rasgos y filtrado.

Conjunto de datos	**RankFSP**	**Preprocesamiento + RankFSP**	**Reducción de consultas**
Train1 MSLR-WEB10K	30.000.000	27.030.535	2.969.465
Train1 MSLR-WEB30K	94.595.000	85.237.004	9.357.996

Tabla 6-7: Número de consultas evaluadas en cada conjunto de datos según el esquema utilizado.

Finalmente, todos estos resultados vienen a corroborar las potencialidades descritas en la Sección 6.4.1 acerca del método propuesto de selección de rasgos. El cual, no solamente posibilita la eliminación de ruido, compactación de información, preparar los datasets para que los métodos de L2R alcancen mejoras en el *ranking*, sino además, junto al proceso de filtrado permite una disminución considerable del tiempo de cómputo y la cantidad de consultas a evaluar.

[9]Valor resultante de la expresión (6000-742)*5000.

6.5. Conclusiones

En este capítulo proponemos un novedoso método de L2R que denominaremos RankFSP. Este método está basado en una variante adaptada y mejorada de FSP y es capaz de construir una función de *ranking* sobre un determinado conjunto de entrenamiento optimizando directamente cualquier medida de evaluación de RI. Esta variante contempla dos pilares principales: una estrategia de lanzamientos condicionados al contexto y una estrategia de reinicios de puntos de captura.

Además, con el objetivo de mejorar la tarea de *ranking* que realizan los métodos de L2R sobre las colecciones de datos analizadas, se propuso también un método simple pero eficaz de selección de rasgos. Posteriormente, se realizó un estudio experimental donde se evalúo el rendimiento de RankFSP y su efectividad en comparación con otros métodos referenciados en la literatura. Durante esta fase experimental, implementamos y evaluamos también, a modo exploratorio, la aplicación de la heurística Supernova al problema del L2R, cuyo método resultante fue etiquetado como RankSupernova.

Todos estos resultados obtenidos fueron procesados estadísticamente siguiendo un análisis de varianza (ANOVA) de 2 vías de Friedman por rangos, incluyendo las comparaciones múltiples por parejas, y usando un intervalo de confianza del 95 %.

Los resultados finales tanto de la fase de evaluación, como del análisis estadístico muestran como RankFSP es un método muy competitivo que alcanza los mejores valores de precisión en el ranking en comparación con los demás algoritmos estudiados, y demuestra además, cuán eficiente resulta el método de selección de rasgos para propiciar mejoras en el rendimiento de métodos de L2R frente a estas colecciones de datos.

Por otro lado, también se llevó a cabo un análisis del coste computacional donde se pudo determinar que la aplicación del método de selección de rasgos y el filtrado sobre la colección MSLR-WEB10K redujo a un 65,81 % el tiempo de entrenamiento de RankFSP, y a un 71,55 % dicho tiempo sobre MSLR-WEB30K.

Finalmente, todos estos resultados muestran las potencialidades de RankFSP para abordar la tarea del L2R, y reafirman las ventajas de utilizar el método propuesto de selección de rasgos para lograr elevar las prestaciones de los métodos de L2R, disminuyendo adicio-

nalmente el tiempo de entrenamiento y la cantidad de consultas a evaluar.

PARTE IV

Herramienta software para la tarea del L2R

Capítulo 7

L2RLab: Entorno Integrado de Experimentación para el Aprendizaje de la Ordenación

"Para demostrar algo hay que servirse de experimentos y reflexiones...".
— Paracelso

7.1. Introducción

En el presente capítulo presentamos la herramienta *Learning-to-Rank Laboratory* (L2RLab) [138], que tiene como propósito fundamental asistir y facilitar la labor del investigador en el proceso de experimentación con nuevos y referenciados modelos de aprendizaje de la ordenación, medidas de evaluación y colecciones de datos.

L2RLab permite el preprocesado individual y por lotes de las colecciones de datos, estudio de la influencia de los rasgos en la ordenación, conversión de formatos a tipos conocidos (por ejemplo, ".arff" de Weka) y visualización. Facilita la comparación de dos o más métodos en cuanto a rendimiento (precisión) alcanzado en la ordenación, incluyendo funcionalidades para realizar análisis estadísticos sobre los rendimientos obtenidos a nivel de colección y/o consulta. Permite además, el estudio del comportamiento de las curvas de aprendizaje de los diferentes métodos de aprendizaje.

L2RLab está programado en Java y está diseñado como una herramienta orientada a la extensibilidad por lo que añadir nuevas funcionalidades es una tarea sencilla. Cuenta además, con una interfaz muy fácil de usar que evita tener que programar las aplicaciones para nuestros experimentos.

Básicamente, L2RLab está formada por dos módulos principales: la aplicación visual y un framework que posibilita la inclusión de los nuevos algoritmos y medidas de rendimiento desarrollados por el investigador.

7.2. Justificación de la herramienta

Actualmente, una gran parte de los trabajos investigativos en el campo del L2R resultan en nuevos modelos y métodos que requieren obligatoriamente de un alto nivel de experimentación para su concepción y refinación.

Normalmente estos experimentos se llevan a cabo empleando una variedad de colecciones de datos y diferentes medidas de evaluación utilizadas en la RI. Incluido a esto, por lo general estos métodos de L2R son parametrizables, es decir, que resulta complejo fijar la combinación ideal de parámetros para una colección determinada.

Además, para demostrar que una propuesta determinada es superior a otras (en rendimiento, tiempo computacional, etc.) es preciso compararlas en situaciones similares, (por ejemplo, requerimientos de hardware, número de evaluaciones, colección de datos, medida de rendimiento, etc.), lo cual no siempre resulta sencillo. Esto se debe principalmente a que los algoritmos según sus autores varían en estructura y lenguajes de programación, entre otras características; y que para la mayoría de los casos resulta inaccesible utilizar los recursos ya implementados por dichos autores.

En los últimos años se han desarrollado diferentes herramientas para el ML que pueden ser adaptadas para asistir a los investigadores dentro de esta rama de la Inteligencia Artificial. Ejemplo de ello es Weka [139], que resulta en una extensa colección de algoritmos de ML desarrollados por la Universidad de Waikato (Nueva Zelanda) implementados en Java y bajo la licencia GPL[1]. A pesar de que Weka contiene las herramientas necesarias para realizar transformaciones sobre los datos, tareas de clasificación, regresión, clustering, asociación y visualización, no contempla un esquema apropiado para la tarea del aprendizaje de la ordenación. En este sentido, resulta más viable y menos costoso diseñar una nueva herramienta, que modificar la estructura de Weka para incluir herramientas que faciliten el L2R.

Por otro lado tenemos aplicaciones sofisticadas como MatLab[2], que constituye un lenguaje de alto rendimiento para la computación técnica, integrando cómputo, visualización, y programación en un ambiente fácil de usar.

[1]GNU Public License. http://www.gnu.org/copyleft/gpl.html
[2]Disponible en: http://www.mathworks.com

También pudiésemos pensar en *Java Data Mining* (JDM). JDM es una interfaz de programación de aplicaciones (API, por sus siglas en inglés) estándar de Java para desarrollar aplicaciones y herramientas de minera de datos.

Sin embargo, ninguna de estas dos alternativas están desarrolladas específicamente para el L2R, de modo que el usuario, si las utilizara, debe consumir un tiempo considerable en su adaptación y puesta en marcha. Incluso, algunas de ellas no son de libre distribución según la licencia GNU[3], como es el caso de MatLab.

Por otro lado, en el sitio web oficial de LETOR[4] (*LEarning TO Rank for Information Retrieval by Microsoft Research*) se brinda todo un esquema de experimentación para facilitar la tarea del L2R, se publican colecciones de datos, los resultados de algunos de los principales métodos que establecen el estado del arte y algunas herramientas de evaluación programadas en lenguaje Perl. Sin embargo, tales elementos resultan insuficientes para llevar a cabo tareas de preprocesamiento, análisis de colecciones, evaluación, comparación y estudio a profundidad del comportamiento de nuevos modelos de L2R, entre otras tareas afines.

Mucho más cercano a tales tareas de L2R resulta la librería RankLib propuesta por Van Dang en noviembre de 2010. Esta librería de algoritmos de L2R, en su última versión 2.1[5] tiene implementado ocho de los algoritmos más populares del área, a decir, MART [140], RankNet [18], RankBoost [17], AdaRank [9], Coordinate Ascent [141], LambdaMART [142], ListNet [67] y Random Forests [143]. Además, implementa varias métricas de evaluación y proporciona desde la línea de comandos diversas formas para realizar la evaluación de modelos. RankLib está disponible bajo la licencia BSD[6] y ahora es parte de *Lemur Project*[7]. Sin embargo, esta herramienta no proporciona funcionalidad alguna para el preprocesamiento y análisis de las colecciones de datos, el comportamiento de las curvas de aprendizaje, así como para el análisis estadístico de los rendimientos obtenidos por cada algoritmo de L2R.

[3]GNU Public License. http://www.gnu.org/copyleft/gpl.html
[4]Disponible en: http://research.microsoft.com/en-us/um/beijing/projects/letor/
[5]Disponible en http://people.cs.umass.edu/∼vdang/data/RankLib-v2.1.tar.gz
[6]http://people.cs.umass.edu/∼vdang/ranklib_license.html
[7]Disponible en https://sourceforge.net/projects/lemur/

En respuesta a estas limitantes existentes, proponemos un entorno integrado de experimentación para el L2R. Esta herramienta denominada L2RLab, facilita la labor de desarrollo de nuevos modelos de L2R a investigadores del área, pues permite preprocesar y analizar colecciones de datos, evaluar y comparar rendimientos de algoritmos, analizar estadísticamente resultados de precisión, estudiar las curvas de aprendizaje de los nuevos métodos, entre otras tareas de interés. En las siguientes secciones se describen las características de los dos módulos principales de L2RLab, así como sus potencialidades para afrontar la tarea del L2R.

7.3. Diseño de experimentos

Experimentar significa variar deliberadamente las condiciones habituales de trabajo para encontrar mejores maneras de proceder y ganar al mismo tiempo un conocimiento más profundo sobre el comportamiento de productos y/o procesos [144]. En muchos campos del saber los investigadores realizan experimentos, por lo general para descubrir algo acerca de un proceso o sistema en particular. Desafortunadamente, en muchas ocasiones estos experimentos se realizan sin un diseño previo, lo que impide obtener todos los resultados potenciales. Es por eso que los diseños de experimentos resultan más útiles que un simple experimento.

7.3.1. Características

Por Diseño de Experimentos se entiende aquella "estrategia de planificación de experimentos tal que las conclusiones válidas y relevantes puedan ser eficientes y económicamente enriquecidas" [145]. En este sentido, se aprecia que el objetivo es obtener conclusiones válidas y relevantes con eficiencia y economía, utilizando como vía para ello la realización de varios experimentos de forma iterativa, pero que hayan sido previamente planificados en la medida de lo posible. También se le define habitualmente como una prueba o serie de pruebas donde se inducen cambios deliberados en los factores para observar y analizar los cambios en la respuesta de salida [146].

A continuación se relacionan algunos de los conceptos relacionados con el Diseño de Experimentos:

- Respuestas. Es el nombre genérico que se da a la característica estudiada. Representa la salida de un experimento.
- Factores. Se designa de esta forma a las variables que se considera puede afectar a la respuesta, y por tanto, se incluyen en el plan de experimentación.
- Niveles. Son los valores que toma un factor en un determinado experimento.
- Tratamiento. Combinación de variantes y/o niveles de cada factor que se utiliza en una determinada prueba o experimento.

7.3.2. Procedimiento

La metodología del diseño de experimentos estudia cómo realizar comparaciones lo más homogéneas posibles, para aumentar en consecuencia, la probabilidad de detectar cambios o identificar variables influyentes, o lo que es lo mismo, estudia el efecto que sobre una variable respuesta tienen un conjunto de otras variables que se pueden llamar variables experimentales, factores o tratamientos. En la literatura es posible encontrar numerosas metodologías para llevar a cabo diseños de experimentos, como por ejemplo las que se plantean en los trabajos de Montgomery [147] y Bartz-Beielstein [148]. Sin embargo, por su fácil adaptación a la experimentación en el área del L2R se decidió seleccionar para esta investigación la propuesta por Delgado Fernández [145], que se muestra seguidamente:

1. Comprensión y formulación del problema. Consiste en la determinación del problema a resolver y su formulación. Es necesario desarrollar todas las ideas sobre los objetivos del experimento.
2. Elección de factores y niveles. Se eligen los factores que variarán en el experimento, los intervalos de dicha variación y los niveles específicos a los cuales se hará el experimento. Debe considerarse la forma en que se controlarán los factores en los valores deseados y como se les medirá.
3. Selección de la variable respuesta. Se debe estar seguro que la respuesta que se va a medir realmente suministre información útil acerca del proceso de estudio.

4. Elección del diseño experimental. Aquí es muy importante tener en cuenta los principios básicos del diseño de experimento y el objetivo experimental. Estos elementos definen el método estadístico a seleccionar.

5. Realización del experimento. Debe hacerse de acuerdo a lo planificado, con las responsabilidades asignadas según las necesidades de tiempo y recursos establecidas.

6. Análisis de los datos. Se utiliza el método estadístico escogido. Generalmente se utilizan paquetes estadísticos incluidos en algún software.

7. Conclusiones y recomendaciones. A menudo es útil usar métodos gráficos para la presentación de resultados y las ejecuciones de seguimiento y pruebas de confirmación para validar las conclusiones del experimento.

El software a desarrollar en este trabajo asistirá al investigador en los pasos 1, 2, 3, 5, 6 y 7 del procedimiento antes mencionado. Para ello, deberá contener colecciones de datos debidamente etiquetadas y algoritmos que faciliten la comprensión del problema, así como permitir la selección y variación de los parámetros relacionados, los cuales constituyen los factores y niveles del experimento. Para la selección de la variable respuesta, L2RLab se basará en las principales medidas de evaluación utilizadas en RI [43, 44]. Finalmente, la mayor responsabilidad de la aplicación recae en la realización automática del experimento. De manera que la elección del diseño experimental, el análisis de los datos, y las conclusiones y recomendaciones quedarán por parte del usuario.

7.4. Aspectos técnicos de L2RLab

L2RLab fue programada sobre la tecnología Java, la cual se basa en un lenguaje de alto nivel desarrollado por la compañía Sun Microsystems[8] (adquirida actualmente por la compañía Oracle[9]). Por lo tanto esta herramienta propuesta es independiente de la arquitectura y funciona en cualquier plataforma sobre la que esté disponible una máquina

[8]http://www.sun.com/
[9]http://www.oracle.com/

virtual Java. Como se mencionaba en la introducción, L2RLab está formada por dos componentes principales: un *framework* para la creación e inclusión de algoritmos de L2R y medidas de evaluación, y una aplicación visual para controlar la ejecución de los experimentos y visualizar los resultados obtenidos, por lo que el usuario sólo debe preocuparse por la programación de sus propuestas. En las secciones siguientes se explican los detalles de estos dos componentes.

7.4.1. Framework

Un Framework Orientado a Objetos (FOO) es el diseño reusable de un sistema que describe cómo debe descomponerse el mismo en un conjunto de objetos que interactúan. A diferencia de una simple arquitectura de software, un FOO está expresado en un lenguaje de programación y está basado en el dominio de un problema específico [149].

Básicamente, un FOO está compuesto por dos tipos de elementos principales: puntos calientes y puntos congelados. Los calientes representan código extensible a través de clases abstractas e interfaces, mientras que los congelados son funcionalidades que no pueden ser cambiadas por el usuario final y que definen la lógica del dominio del problema (en este caso, la experimentación con modelos de L2R) [146]. En ese sentido, con la idea de proveer al investigador de las facilidades necesarias en la programación de propuestas y problemas se implementó un FOO, cuyas clases se muestran en la Figura 7-1.

Las clases con una tonalidad más intensa representan los puntos calientes (por ejemplo, aModel, aCollection, etc.), mientras que las más claras son los puntos congelados. En este diagrama se han excluido otros métodos y atributos para una mejor comprensión.

Note que un experimento, definido por la clase L2RExperiment, consta de uno o muchos modelos de L2R y de una o varias colecciones de datos, donde esta clase es la encargada de llevar a cabo y controlar las diversas tareas del L2R. Existe además una clase denominada EvaluationMeasures, que reúne una colección de medidas de rendimiento encargadas de cuantificar el rendimiento de los modelos de aprendizaje con respecto a las colecciones de datos utilizadas.

La inclusión de interfaces permite más libertad al usuario a la hora de programar sus propuestas, mientras que las clases abstractas les reduce el tiempo de implementación debido

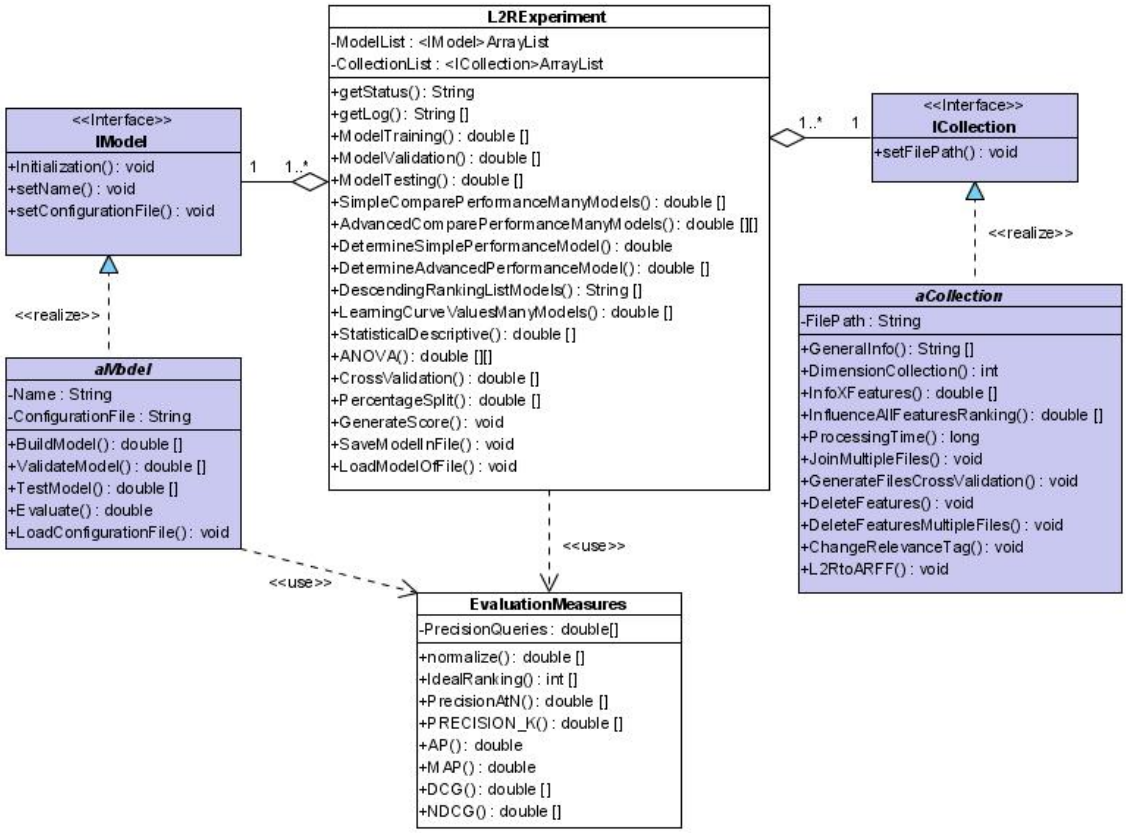


Figura 7-1: Diagrama de clases del *framework* incluido en L2RLab - Algunos métodos, atributos y parámetros son excluidos para una mejor comprensión.

a que estas contienen ya ciertas funcionalidades, de manera que el usuario puede extender el *framework* a partir de las interfaces o de las clases abstractas. La clase L2RExperiment es la encargada finalmente de llevar a cabo la tarea del L2R.

7.4.2. Aplicación visual

La aplicación visual desarrollada es simple e intuitiva. A través de tres módulos visualmente conectados desde una misma ventana principal es posible acceder a las distintas funcionalidades del sistema.

Los módulos del sistema son los siguientes: (1) Pre-procesamiento de Colecciones de Datos, (2) Entrenamiento y Prueba de Modelos, y (3) Comparación y Análisis de Modelos. En la Figura 7-2 se visualiza la pantalla principal de L2RLab, donde se muestra una breve descripción y un vínculo de acceso a cada uno de estos módulos en divisiones de una misma

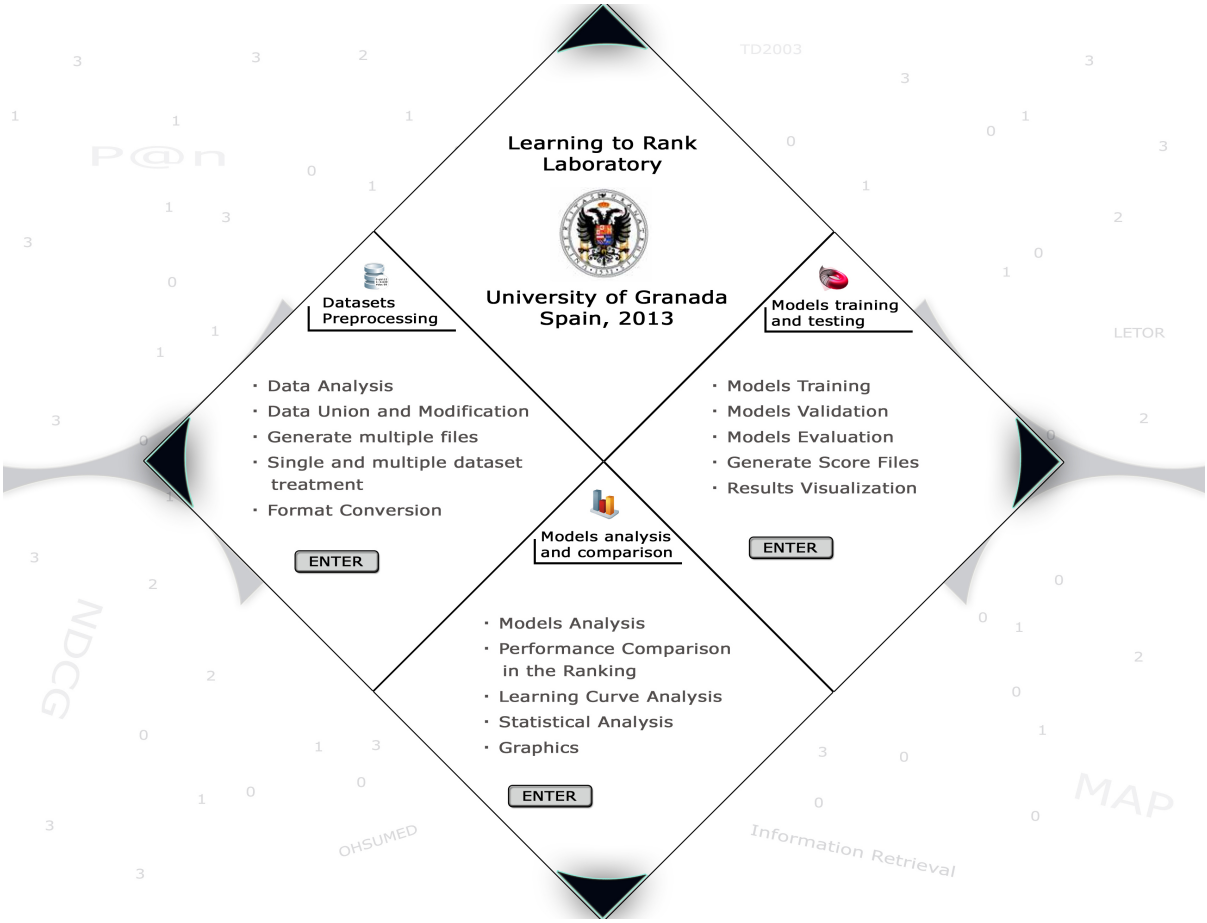


Figura 7-2: Interfaz principal de la aplicación L2RLab.

interfaz.

La clase principal de la aplicación se comunica con el código del usuario a través del *framework*, específicamente con la clase L2RExperiment. Cada modelo de L2R está representado por una clase implementada por el usuario, que tiene una contraparte en un fichero de configuración que es cargado (véase el método LoadConfigurationFile) cuando inicia la aplicación.

Además de las definiciones de los parámetros del algoritmo de aprendizaje, en estos ficheros se incluye el nombre de la clase con una breve descripción de la misma. Cada clase definida por el usuario recibe a través de su constructor los valores de los parámetros propios del algoritmo, así como otros específicos de las ejecuciones. La Figura 7-3 muestra un ejemplo

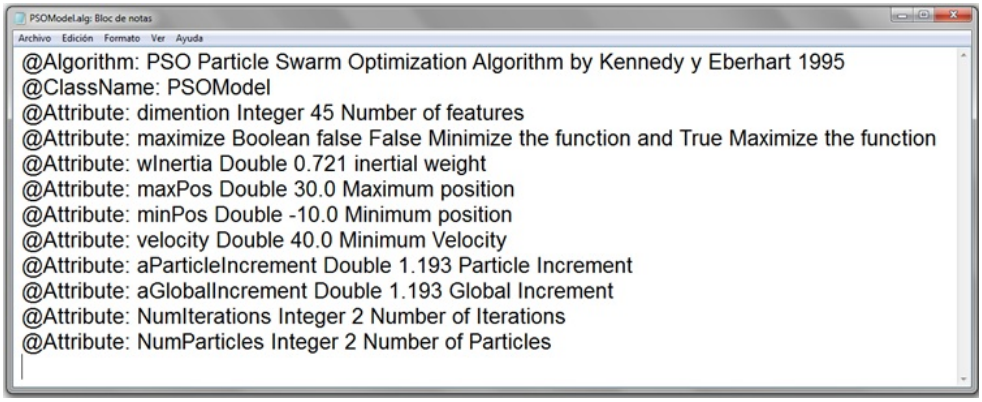


Figura 7-3: Fichero de configuración en L2RLab. Cada método implementado está asociado con un fichero similar, el cual permite establecer valores a los atributos de la clase sin tener que modificar el código fuente.

de la estructura de este fichero de configuración.

Una de las ventajas que brinda la inclusión de ficheros de configuración como los empleados, es que cada clase puede estar asociada a varios ficheros distintos, con la consecuente derivación de varias instancias o escenarios por cada algoritmo de aprendizaje.

7.4.3. Requerimientos de Hardware

Los requerimientos mínimos recomendados en cuanto a memoria de acceso aleatorio (en inglés: *random-access memory*) y velocidad del procesador para instalar L2RLab en una estación de trabajo, están condicionados sencillamente a poder instalar una versión igual o mayor a la versión 6.0 de la Máquina Virtual de Java[10].

Por otro lado, el tamaño mínimo disponible en disco duro está dado por el paquete software en general y por las colecciones de datos a utilizar. La Tabla 7-1 muestra un resumen del tamaño y otras características de las principales colecciones de datos que se emplean actualmente para llevar a cabo la tarea del L2R por los investigadores del área.

7.5. Principales funcionalidades

En las siguientes subsecciones se explican las principales funcionalidades de L2RLab, específicamente de la parte visual.

[10]Disponible en http://www.java.com/java6download

Tabla 7-1: Resumen de las principales colecciones de datos utilizadas en el área del L2R.

Referencia	Colección	Conjunto de Datos	Tamaño	Liberada
LETOR 3.0	- OHSUMED	OHSUMED	≈ 166 MB	Diciembre, 2008
	- Gov	HP2003	≈ 558 MB	
		HP2004	≈ 283 MB	
		NP2003	≈ 562 MB	
		NP2004	≈ 281 MB	
		TD2003	≈ 185 MB	
		TD2004	≈ 282 MB	
LETOR 4.0	- Gov2 - TREC 2007 - TREC 2008	MQ2007	≈ 370 MB	Julio, 2009
		MQ2008	≈ 81.2 MB	
Microsoft Learning to Rank Datasets	- Commercial web search engine (Microsoft Bing)	MSLR-WEB10K	≈ 6.5 GB	Junio, 2010
		MSLR-WEB30K	≈ 20.3 GB	
		Total	≈ **29.5 GB**	

7.5.1. Conversión y tratamiento de colecciones de datos

El módulo de conversión y tratamiento de colecciones de datos permite realizar diversas operaciones sobre uno o múltiples conjuntos y/ó colecciones de datos.

En cuanto al origen de los datos, L2RLab soporta para el análisis y procesado de las colecciones el formato estándar de los ficheros de datos de L2R; donde cada fila de este fichero representa un par consulta-documento. La primera columna constituye el juicio de relevancia para cada par, la segunda columna es el identificador de la consulta, las siguientes columnas son los rasgos extraídos de cada documento con respecto a dicha consulta; y al final de cada fila se tiene un comentario acerca de cada par (consulta-documento), que siempre incluye el identificador del documento y en algunos conjuntos de datos, otras informaciones de interés. A modo de ejemplo, mostramos a continuación dos filas del conjunto de datos MQ2007:

==========

2 qid:10032 1:0.056537 2:0.000000 3:0.666667 4:1.000000 5:0.067138 ... 45:0.000000 46:0.076923 #docid = GX029-35-5894638

inc = 0.0119881192468859 prob = 0.139842

0 qid:10032 1:0.279152 2:0.000000 3:0.000000 4:0.000000 5:0.279152 ... 45:0.250000 46:1.000000 #docid = GX030-77-6315042

inc = 1 prob = 0.341364

==========

Conocido esto y una vez seleccionado un determinado conjunto de datos, la aplicación nos muestra según la ventana que visualiza la Figura 7-4. la información básica de dicha colección; es decir, la cantidad de consultas de la colección, de rasgos, de documentos irrelevantes y la cantidad de documentos relevantes desglosados según la categoría de relevancia (*ground truth*) a la que pertenecen. Se muestra además, la cantidad de documentos recuperados por cada una de las consultas y por cada uno de los atributos que componen los datos; se visualiza también, un breve resumen de estadísticos descriptivos que incluyen: el rango de los datos, la media aritmética y desviación estándar.

A partir de esta interface, se puede determinar también, la influencia que ejerce cada rasgo en particular para la tarea de la ordenación. En esta acción podemos chequear primeramente el tiempo que demorará realizar tal operación (según las características y el tamaño de la colección y las prestaciones de hardware del ordenador que tengamos). Si ejecutamos tal funcionalidad obtendremos un gráfico de barras verticales, donde se refleja cuán influyente es cada rasgo en el *ranking*, además de facilitarnos en una nueva ventana cada uno de los rasgos ordenado descendentemente según su fortaleza o disposición para alcanzar una mejor ordenación sobre dicha colección de datos.

Además de permitirnos conocer información pertinente de una o más colecciones, este primer módulo de L2RLab nos brinda la posibilidad de realizar otras tareas a nivel individual o por lotes, como por ejemplo, la unión y modificación de archivos de datos, el agrupamiento de conjuntos de datos a partir de colecciones generales, la conversión de formatos, entre otros.

La modificación de uno o más ficheros de datos (ver interface en la Figura 7-5), puede ser realizada a través de diferentes opciones que posibilitan la eliminación individual o grupal, tanto de consultas y rasgos, como de documentos. Para el caso de los juicios de relevancia, pueden ser sustituidos por nuevas etiquetas.

L2RLab da la posibilidad al usuario de especificar nombres a los ficheros resultantes de cada operación, en caso de no hacerlo, la aplicación utiliza un convenio de nombres intuitivos

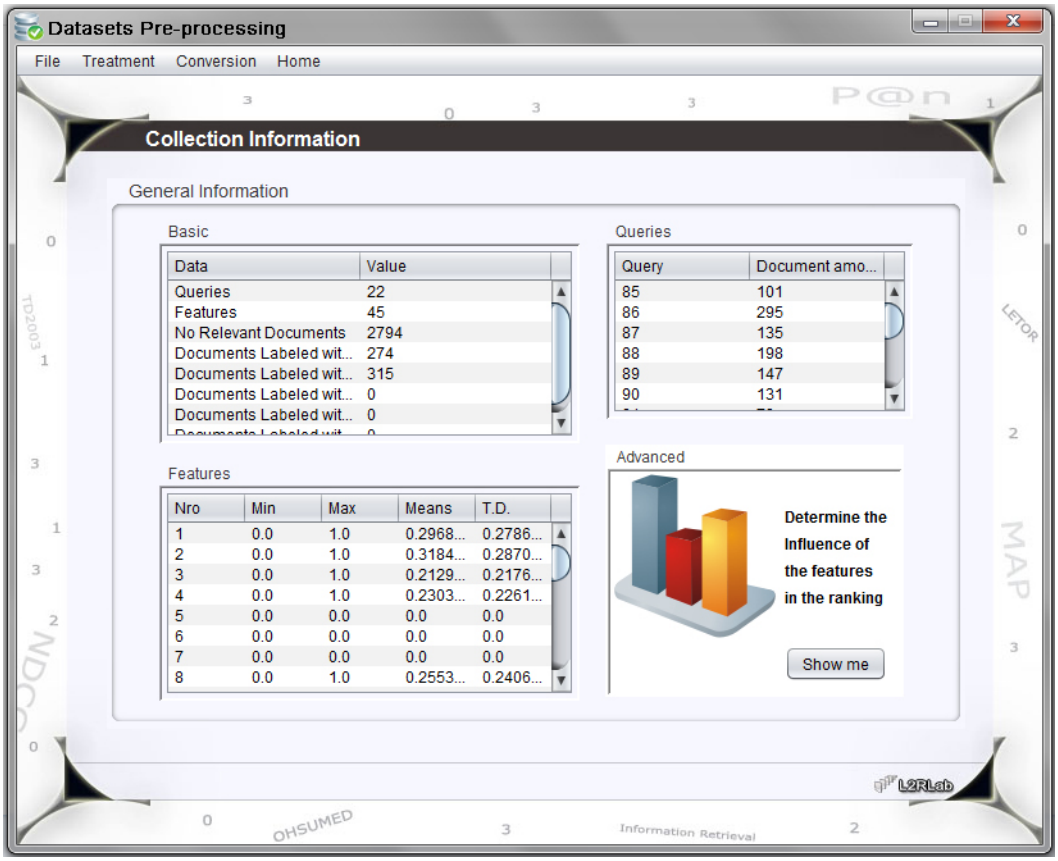


Figura 7-4: Pantalla del módulo n^o.1 que muestra información específica de una colección de datos.

por cada acción. Por otro lado, permite salvar en nuevos ficheros las modificaciones y eliminaciones realizadas sobre las colecciones, manteniendo inalterables los datos originales y garantizando la fiabilidad de un nuevo preprocesamiento de los datos.

Otra de las características comunes para todas las funcionalidades en cada una de las interfaces es la de utilizar una barra de progreso y notificar el estado y resultado de las diferentes operaciones, siendo accesible dicha información a través del botón *Log*.

Para el caso típico de unión de dos conjuntos de datos, la herramienta también nos da opciones para especificar cuáles consultas serán consideradas y como quedará estructurada la información en el nuevo fichero resultante.

En la Figura 7-6, se aprecia como la herramienta también es capaz de generar de forma controlada múltiples ficheros de datos, a partir de la información de un fichero principal, donde se tienen todas las consultas pertenecientes a una determinada colección de datos.

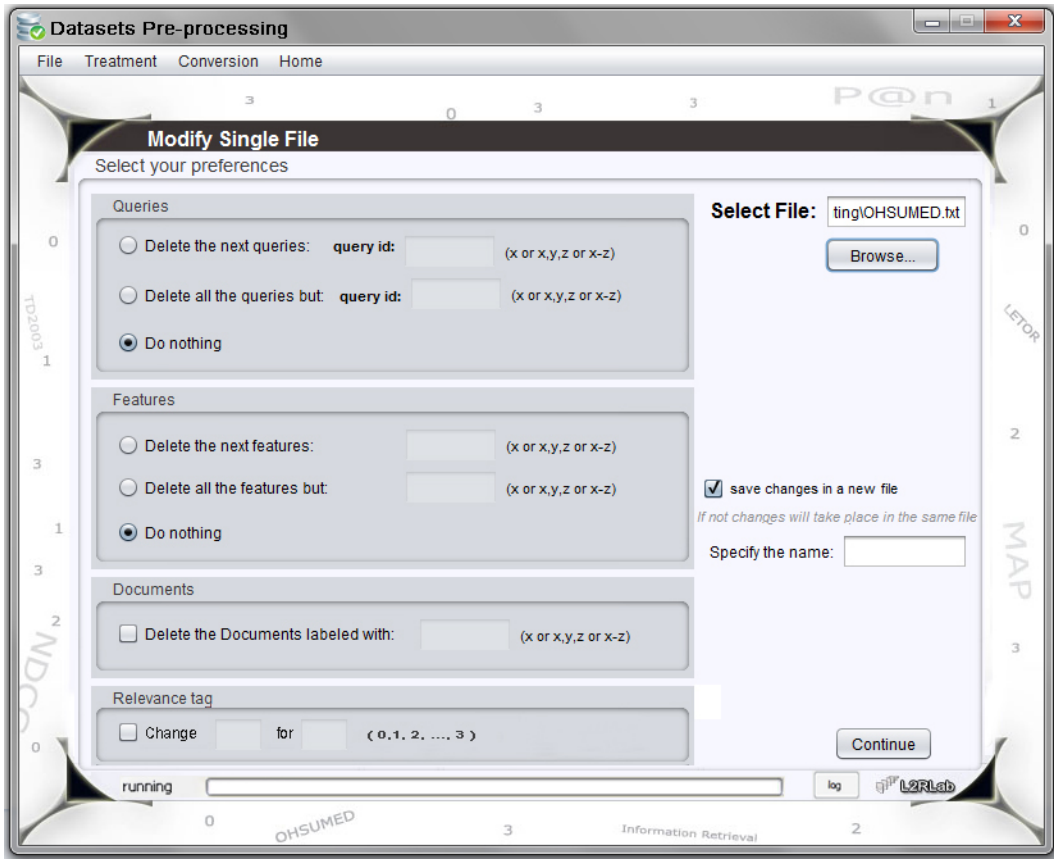


Figura 7-5: Pantalla del módulo n^o.1 que muestra las opciones posibles para modificar la información de una colección de datos.

En un principio se tiene la opción de crear tantos ficheros como consultas tenga la colección. En una segunda opción es posible generar los tres ficheros clásicos para la experimentación (entrenamiento, validación y prueba), al especificar qué porcentaje de los datos del fichero principal, le corresponderá a cada uno de estos nuevos ficheros.

Finalmente, el investigador, también tiene la opción de decidir qué consultas del fichero principal serán incluidas en la conformación de los ficheros de entrenamiento, validación y prueba.

En estos casos, se debe ser cuidadoso para no potenciar una fractura de datos y, en sentido general, recomendamos utilizar el esquema experimental (*5-fold-cross validation*) prefijado en LETOR para cada *dataset* a partir de cada colección.

Por otra parte, a través de una interface sencilla, L2RLab nos permite seleccionar y convertir un determinado conjunto de datos de L2R al formato conocido de Weka (.arff).

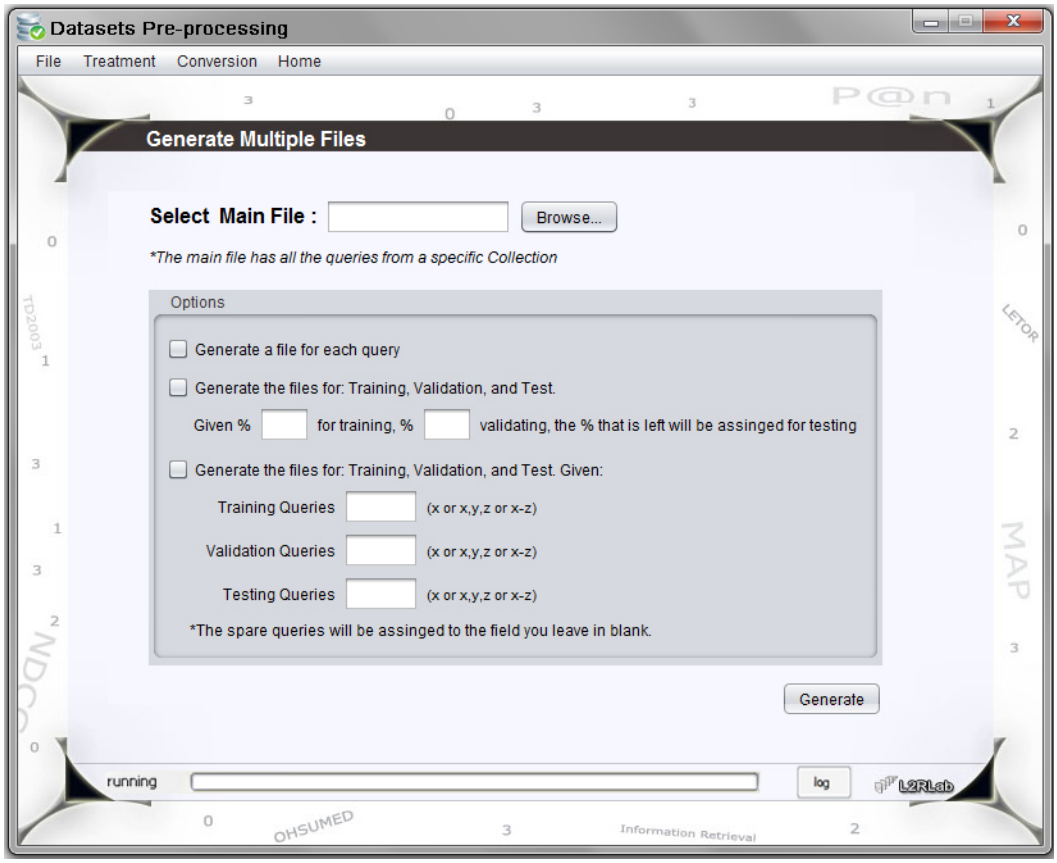


Figura 7-6: Pantalla del módulo n^o.1 que muestra las opciones para generar múltiples ficheros a partir del fichero principal de una colección de datos.

En el fichero transformado, los rasgos serán considerados como atributos, donde aquellos rasgos perdidos (*missing*) serán etiquetados con el carácter "?"; y por su parte, los juicios de relevancia serán las clases.

7.5.2. Entrenamiento y prueba de modelos

Como se puede apreciar en la Figura 7-7, esta pantalla permite el control del entrenamiento y evaluación de nuevos modelos de L2R.

En un primer momento, se selecciona el algoritmo de aprendizaje, el conjunto de entrenamiento y la medida de evaluación utilizada para construir la función de *ranking*.

Luego, la herramienta, permite especificar una serie de opciones para llevar a cabo la evaluación del modelo aprendido: (1) utilizar enteramente el mismo conjunto de entrenamiento, (2) seleccionar un nuevo conjunto de datos, (3) realizar una validación cruzada

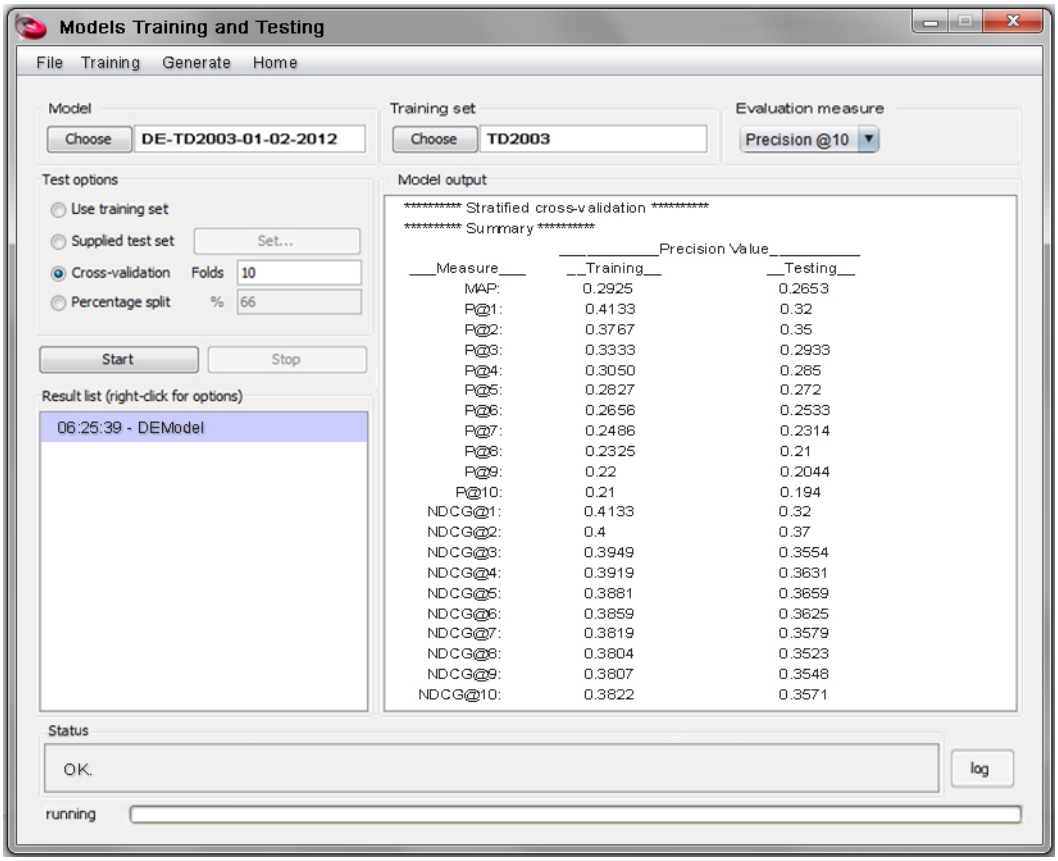


Figura 7-7: Pantalla del módulo n^o.2 que permite realizar entrenamientos para evaluar el comportamiento de los modelos de L2R.

de n particiones, donde el valor de n será prefijado por el investigador, y finalmente, (4) especificar qué porcentaje del conjunto de entrenamiento será utilizado para la evaluación. Una vez terminada la fase de entrenamiento y prueba, se visualiza en el área de resultados, el modelo aprendido, el tiempo computacional empleado para construir el modelo y un resumen del rendimiento alcanzado por el algoritmo para las medidas de evaluación más usadas en el L2R para la RI, a decir, *Mean Average Precision* (MAP) [43], *Normalized Discounted Cumulative Gain* (NDCG) [44] y *Precision at n* (P@n) [43]. Para estas dos últimas medidas se muestra la precisión en las 10 primeras posiciones del *ranking*.

Cada una de estas pruebas realizadas al modelo, no se eliminan, sino que se van registrando en la lista de resultados (ubicada en la parte inferior izquierda). Esta opción tiene la ventaja de poder ir consultando los resultados anteriores, mientras se ejecutan otras pruebas. Estos resultados por demás, se guardan en ficheros de texto independiente dentro de la carpeta

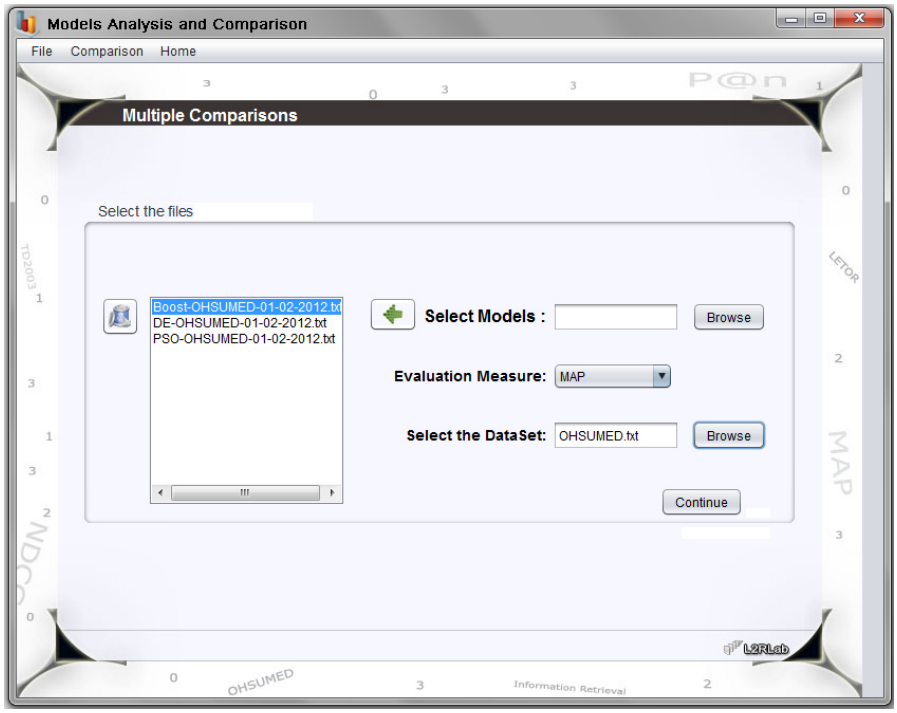


Figura 7-8: Pantalla del módulo n^o.3 que posibilita seleccionar la medida de evaluación, el conjunto de datos, y los modelos de aprendizaje que serán comparados.

"Resultados" del directorio principal de L2RLab.

El panel inferior de esta pantalla se brinda información actualizada del estado de la ejecución actual del algoritmo y haciendo click en el botón *log* se podrá acceder al historial de sucesos del sistema para este módulo.

Otra de las funcionalidades de este módulo radica en crear un fichero de puntuaciones (*scores*) a partir de un modelo de L2R y un conjunto de datos. Estas puntuaciones representan cuan relevantes considera la función de *ranking* a un determinado documento recuperado con respecto a la consulta correspondiente.

Estos ficheros de puntuaciones pueden ser utilizados para determinar el rendimiento del método evaluado a nivel de consulta.

7.5.3. Comparación y análisis de modelos

El tercer módulo de L2RLab persigue analizar y comparar el rendimiento de los diferentes métodos de L2R, que sean de interés para el investigador.

Partiendo del formulario que refleja la Figura 7-8, podemos configurar nuestro propio

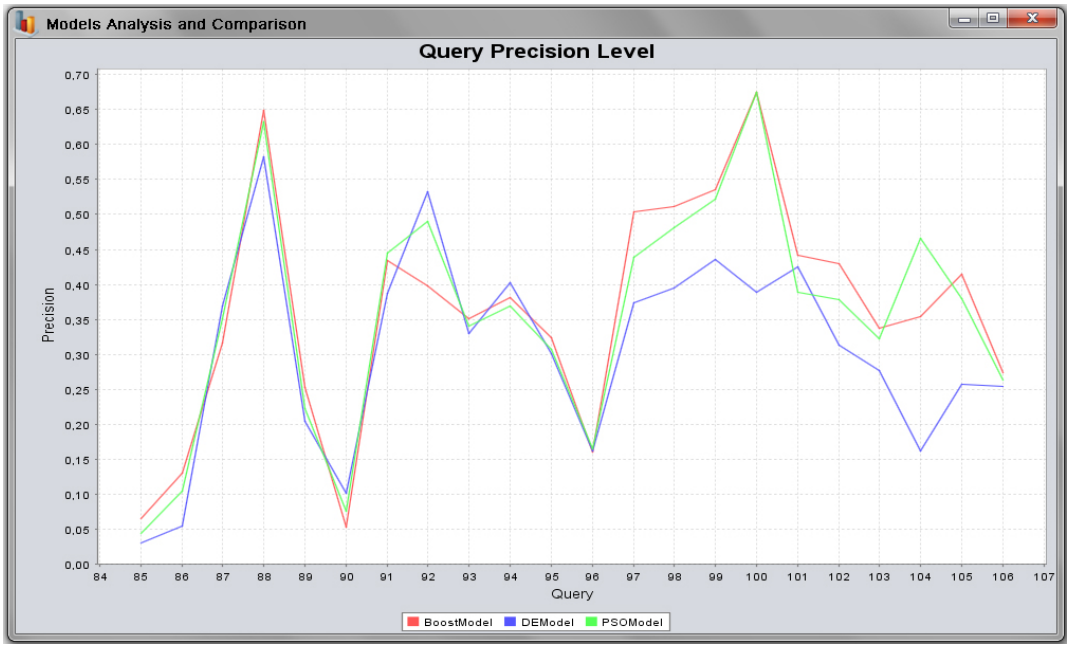


Figura 7-9: Pantalla del módulo n^o.3 que visualiza la precisión en el *ranking* alcanzada por múltiples modelos de ordenación a nivel de consulta.

esquema de comparación para determinar el rendimiento alcanzado por cada método de aprendizaje en cuanto a precisión en el *ranking*. Seleccionamos entonces, los métodos que vamos a confrontar, la medida de evaluación y el conjunto de datos correspondiente.

Los resultados finales son visualizados de forma gráfica, donde se muestran los rendimientos obtenidos por cada método a nivel de conjunto de datos y por cada consulta (ver Figura 7-9) en específico. La evaluación a nivel de consulta resulta vital para determinar cuán robusto y estable resulta el modelo aprendido por cada algoritmo.

Otras de las opciones funcionales de este módulo está encaminada a facilitar el estudio del comportamiento de las curvas de aprendizaje de cada método de L2R. En este sentido, la gráfica que muestra la Figura 7-10, nos da la posibilidad de observar con el paso de n iteraciones cómo se comporta el aprendizaje de cada algoritmo al ir construyendo su función de *ranking* correspondiente. Este estudio resulta vital para comprender el funcionamiento de un método de aprendizaje y llegar a conocer su capacidad de adaptación bajo diversas condiciones de entrenamiento.

Por otro lado, L2RLab permite realizar pruebas no paramétricas. Específicamente, en la

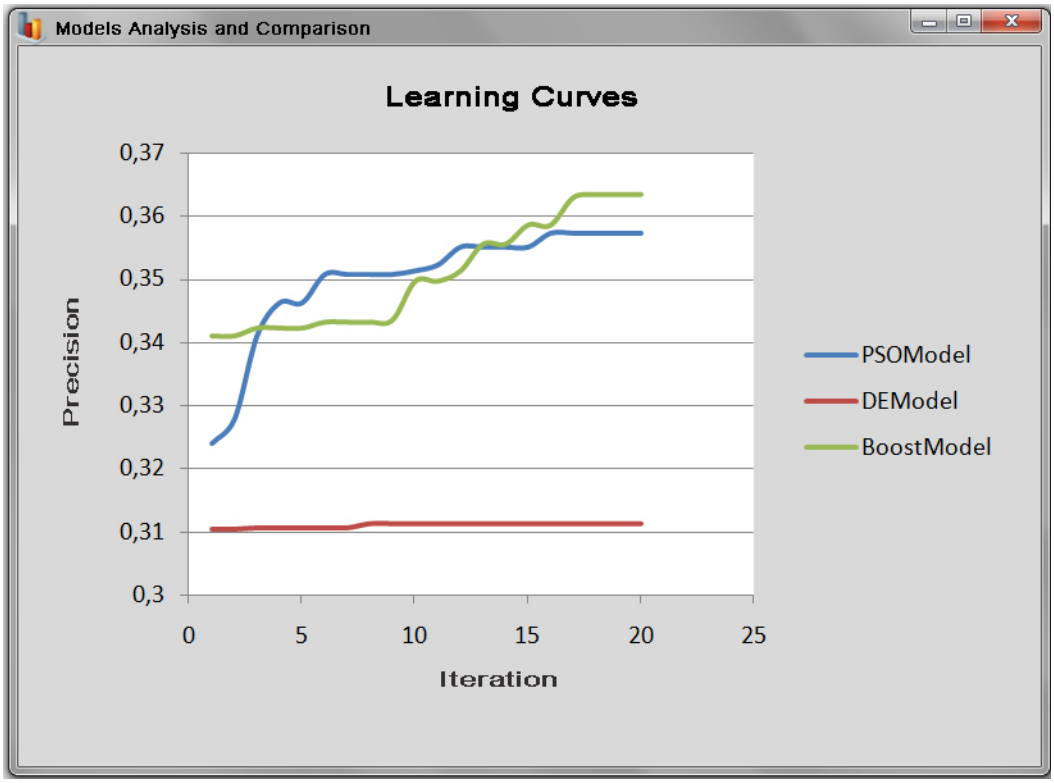


Figura 7-10: Pantalla del módulo n^o.3 que visualiza para n (=20) iteraciones el comportamiento de las curvas de aprendizaje de múltiples modelos de L2R.

pantalla de la Figura 7-11, se muestra todo un esquema para llevar a cabo un análisis de varianza de segunda vía (two way ANOVA) [113] [150]. Con esta intención, la prueba de Friedman puede ser usada [114, 115]. Si la hipótesis nula es rechazada, podemos proceder entonces con una prueba *post-hoc*. En este caso, se podrá utilizar la prueba de Nemenyi [151] [150], que es similar a la prueba de Tukey para el ANOVA y es usada cuando todos los algoritmos son comparados entre sí.

Este modelo es propuesto porque se ajusta bien a la necesidad de comparar diferentes algoritmos en un mismo grupo de consultas, todo lo cual permite llevar a cabo un estudio comparativo de los rendimientos de cada algoritmo con un alto nivel de veracidad y exactitud.

Una vez concluido el análisis, se muestra un reporte completo de los principales resultados obtenidos, lo que permitirá al investigador conocer las diferencias y el nivel de significación en cuanto a precisión que se establecen entre los diferentes algoritmos de L2R que son

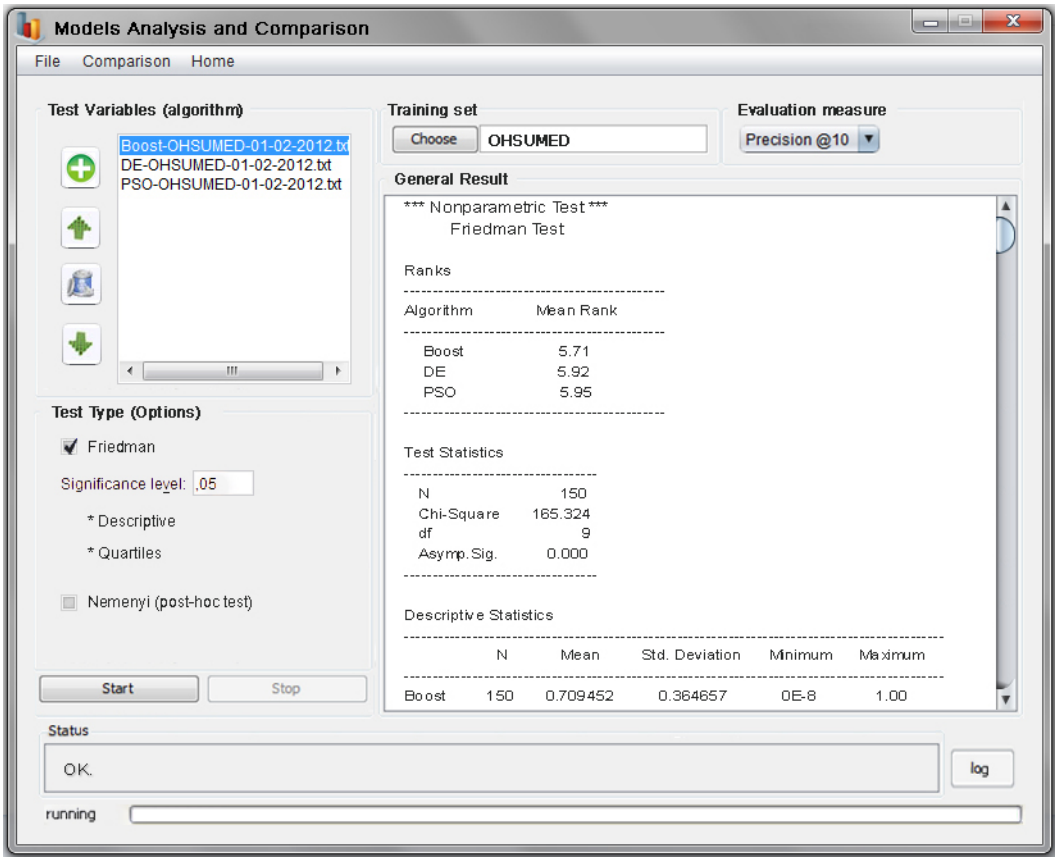


Figura 7-11: Pantalla del módulo n^o.3 que permite realizar pruebas no paramétricas, considerando el rendimiento de los métodos de L2R a nivel de consulta.

estudiados.

7.6. Un caso de estudio

Para mostrar las bondades que ofrece L2RLab, en este apartado describiremos un caso de estudio en particular.

La tarea consiste en incluir un nuevo algoritmo de L2R, basado en Optimización con Enjambre de Partículas (*Particle Swarm Optimization*, PSO) e inspirado en el método RankPSO [100].

Para ello, haremos uso de un paquete escrito en java para la optimización de enjambre de partículas denominado JSwarm[11], el cual fue diseñado de una forma muy modular y para

[11]Available in: http://en.sourceforge.jp/frs/g_redir.php?m=jaist&f=%2Fjswarm-pso%2Fjswarm-pso%2Fjswarm_pso_1_2%2Fjswarm-pso.jar

ser utilizado con el mínimo esfuerzo.

En tanto, para alcanzar un diseño simple de RankPSO, se hace necesario establecer una buena interconexión entre JSwarm y el *framework* diseñado para L2RLab. Para lograr esto, hemos creado una clase denominada *PSOModel*, la cual hereda de la clase abstracta *aModel*.

Asociado a esta clase, creamos un fichero de configuración con los siguientes parámetros: 45 como valor de dimensión (condicionado a la cantidad de rasgos de la colección), maximizar en TRUE, peso inercial 0.721, posición máxima 30.0, posición mínima -10.0, velocidad mínima 40.0, incremento de cada partícula 1.193, incremento global 1.193, número de iteraciones 50 y cantidad de partículas 40. Los valores de ciertos parámetros han sido determinados considerando el trabajo en [98] sobre el comportamiento de PSO en problemas continuos y espacios multidimensionales complejos.

Después de esto y de una forma simple, implementamos la función *LoadConfigurationFile()* de nuestra clase *PSOModel*.

En la clase *MyFitnessFunction* de JSwarm, dentro del método *evaluate()* pasamos a instanciar la clase *EvaluationMeasures* perteneciente a nuestro *framework* y entonces especificamos la medida MAP como función de evaluación (*fitness function*), indicando en el constructor de esta clase que esta función debe ser maximizada.

Por tanto, en la función *Evaluate()* de *PSOModel*, fijamos como resultado de esta función, la salida del método *evaluate()* que está en la clase *MyFitnessFunction*.

Por otro lado, en el método principal *BuildModel()* de nuestro modelo creamos una instancia de la clase *Swarm* a partir del paquete usado. A esta instancia le pasamos como parámetros los valores leídos del fichero de configuración correspondiente; los demás parámetros definidos genéricamente como la dirección al conjunto de entrenamiento serán especificados en la interface visual. Invocamos entonces, el método *evolve()*. Para el caso del método *TestModel()*, nosotros solamente fijamos como parámetros el conjunto de prueba y el nombre de la medida de evaluación; pues este método usa directamente la clase *EvaluationMeasures*.

Una vez concluida la relación entre las clases, pasamos a la interface mostrada en la Figura 7-7, donde especificamos cada una de las opciones necesarias para llevar a cabo nuestro

experimento (entrenamiento y prueba del modelo).

Es importante resaltar que cuando seleccionamos el conjunto de entrenamiento, es creada una clase *OHSUMEDCollection* que hereda de la clase abstracta *aCollection*, y se actualiza también los valores del constructor de la clase *MyParticle* de JSwarm. En este último caso, para definir a partir de la cantidad de rasgos extraídos de cada documento, la dimensión en la que se moverán las partículas.

Siguiendo el esquema experimental de LETOR, llevamos a cabo una validación cruzada de cinco particiones y como resultados finales tenemos los mostrados en la Figura 7-2. Este esquema experimental nos permite llevar a cabo comparaciones directas con aquellos métodos de L2R referenciados en la literatura y cuyos rendimientos han sido publicados en LETOR.

Finalmente, podemos concluir que para todas las medidas de evaluación consideradas, los valores de precisión alcanzados por el modelo [100] implementado son altamente competitivos (el mejor y segundo mejor resultado en todos los casos), tanto en las 10 primeras posiciones del *ranking*, como de forma general si consideramos la medida MAP.

Este ejemplo, constituye un primer paso para futuros trabajos que estarán encaminados a incluir nuevos algoritmos y medidas de evaluación dentro del L2R, con el objetivo de seguir enriqueciendo este nuevo *framework* con los principales exponentes del estado del arte del L2R. Promoviendo la comparación y el análisis estadístico entre las nuevas propuestas, así como el ahorro de tiempo en la codificación y visualización de los resultados.

Tabla 7-2: Rendimientos obtenidos por el modelo tratado sobre la colección de OHSUMED.

MAP									
0.4525									
P@n									
n=1	**n=2**	**n=3**	**n=4**	**n=5**	**n=6**	**n=7**	**n=8**	**n=9**	**n=10**
0.6718	0.6192	0.5926	0.5929	0.5786	0.5577	0.5470	0.5330	0.5211	0.5058
NDCG@n									
n=1	**n=2**	**n=3**	**n=4**	**n=5**	**n=6**	**n=7**	**n=8**	**n=9**	**n=10**
0.5584	0.4940	0.4824	0.4818	0.4784	0.4707	0.4675	0.4607	0.4586	0.4539

La futura disponibilidad de esta herramienta será lo más pronto posible, y será de libre

acceso desde el sitio del Grupo de Investigación de Tratamiento de la Incertidumbre en Inteligencia Artificial[12].

7.7. Conclusiones

En el capítulo propusimos un entorno integrado de experimentación para el L2R, que denominamos L2RLab. Esta herramienta software le permitirá al investigador desarrollar de forma eficiente y eficaz todas las tareas pertinentes durante la experimentación, evaluación, comparación y análisis de nuevos modelos de L2R.

Las principales ventajas que presenta L2RLab son: constituye una herramienta práctica para asistir el proceso de experimentación con modelos de L2R, posibilita la reutilización de códigos y recursos implementados, ofrece una mayor rapidez (ahorro de tiempo) y confiabilidad (reducción de errores) en el proceso de experimentación. Estas condiciones, le permitirán al investigador centrarse en la tarea del L2R, sin tener que preocuparse en demasía por aspectos técnicos en el diseño y preparación de los experimentos. Otro de los aspectos positivos, será su liberación, con lo cual no se necesitará pagar licencias para usarla.

Finalmente, en el último apartado se describe un caso de estudio en particular, con el objetivo de ilustrar como proceder de la manera más sencilla con L2RLab para adicionar y evaluar un nuevo algoritmo de L2R.

[12]Información del grupo disponible en: http://decsai.ugr.es/gte/index_gte.html

PARTE V

Conclusiones y Trabajos futuros

Capítulo 8

Conclusiones y Trabajos futuros

"No creo haber conseguido ya la meta
ni me considero perfecto,
más bien sigo adelante
con la esperanza de alcanzarlo".
— Filipenses 3, 12

8.1. Conclusiones Generales.

En la primera parte de esta memoria investigativa, se puede constatar como los métodos tradiciones de RI, han ido cediendo su espacio a nuevas técnicas de ML, todo ello propiciado por la necesidad imperante de afrontar nuevos retos y problemas de *ranking* que exigen constantemente soluciones y modelos de gran precisión, simplicidad, bajo costo computacional, de poca complejidad algorítmica, aplicables a diferentes entornos, y que por ende, su calidad esté dirigida a mejorar la satisfacción de los usuarios.

En este sentido, el L2R ha venido a desempeñar un rol vital para muchas aplicaciones dentro de la RI, el procesamiento del lenguaje natural y la minería de datos. Específicamente, en esta parte inicial se presentó una revisión sintetizada del estado del arte sobre el L2R para la RI, donde se han presentado y descrito una serie de métodos relevantes de L2R siguiendo un orden histórico y definiendo su desglose categórico. Todo este bagaje informativo nos da una panorámica general de hacia donde se dirigen las principales tendencias en la forma de concebir y construir nuevos modelos de L2R y cuáles suelen ser las vertientes más prometedoras como puntos de partida para futuros trabajos científicos.

En la parte dos de este trabajo, profundizamos un tanto en este conocimiento, al llevar a cabo un estudio bibliométrico de la producción científica sobre el L2R en el período de los años 2000 al 2013. En el que a través de indicadores métricos evaluamos el comportamiento de la actividad científica y tecnológica sobre L2R, así como la estructura y dinámica social de los que producen y utilizan su literatura. Todo lo cual nos permitió tener un conocimiento mucho más sólido del estado actual de este campo multidisciplinar, y contar

con argumentos certeros para iniciar y perfilar la presente investigación. Además, otro de los aportes de este estudio, consistió en agrupar y caracterizar todas las temáticas relacionadas con el L2R.

Específicamente, para la parte tres de la investigación introducimos una serie de modelos y estrategias para mejorar la tarea del L2R. Todas estas propuestas fueron validadas a través de evaluaciones experimentales, comparaciones con métodos referenciados en la literatura, estudios estadísticos y análisis de los costos computacionales; utilizando para ello diferentes medidas de evaluación de RI y diversas colecciones estándares y establecidas para L2R.

Finalmente en la parte cuatro, se presentó una herramienta software que asiste al investigador y le permite desarrollar de forma eficiente y eficaz todas las tareas pertinentes durante la experimentación, evaluación, comparación y análisis de nuevos modelos de L2R.

En lo adelante exponemos las principales conclusiones que fueron obtenidas como resultado del cumplimiento de los objetivos planteados en este trabajo y del análisis de los diversos estudios experimentales. Finalmente llegamos a las siguientes conclusiones:

- El análisis de las tendencias categóricas en L2R y de los modelos más representativos, sugieren que los métodos de *ranking* que llevan a cabo una optimización directa de medidas de evaluación, ofrecen elementos teóricos y resultados prometedores que ostentan mejoras significativas en cuanto al rendimiento alcanzado en la ordenación con respecto a las demás propuestas y estrategias.

- Además de la información valiosa obtenida del análisis de los diversos indicadores del estudio bibliométrico, uno de los principales resultados de este trabajo está dado por la propuesta de clasificación y agrupamiento de todas las temáticas relacionadas con el L2R, que nos da una idea clara de cómo se han ido comportando cada una de ellas desde el año 2000 hasta la actualidad, cuáles se mantienen en auge, las que son tratadas puntualmente, las que resultan más prometedoras o las que han pasado a ser obsoletas o han sido abandonadas en el tiempo.

- Se propuso un nuevo método de L2R que, basado en PSO, es capaz de optimizar directamente cualquier medida de evaluación utilizada en RI. Este algoritmo que

denominamos RankPSO usa un diseño simple a través de una función de *ranking* lineal e implementa una estrategia de diversificación con el objetivo de prevenir convergencias prematuras a mínimos locales.

- Los resultados experimentales confirman que RankPSO mejora significativamente el rendimiento de los principales métodos de L2R referenciados en la literatura, alcanzando siempre la primera o segunda mejor posición en el *ranking* en términos de precisión frente a todas las medidas de evaluación y conjuntos de datos de la colección .Gov de LETOR.

- Se verificó además que para todos los conjuntos de datos y medidas de evaluación, RankPSO alcanza en su rendimiento un coeficiente de variación promedio de 0.95 %; lo cual nos indica que nuestro algoritmo puede ser considerado como estable en estos casos.

- Los rendimientos alcanzados por RankPSO y AdaRank-NDCG demostraron que la aplicación del procedimiento combinado de reducción de dimensionalidad usando PCA y análisis de Clúster, resulta factible y eficaz para la tarea del *ranking* y que no sólo contribuye a mejorar el rendimiento de los métodos de L2R sino que además le permite a los modelos de aprendizaje disminuir el tiempo computacional empleado durante la fase de entrenamiento.

- Se desarrolló y validó una nueva propuesta metaheurística, denominada Procedimiento de Búsqueda del Pescador, que está encaminada a tratar problemas de optimización global.

- FSP mostró ser una metaheurística simple, intuitiva, que por su formulación puede ser aplicada a un gran número de problemas de optimización y que manifiesta un comportamiento eficiente en la búsqueda de soluciones para todos los problemas teóricos evaluados y en comparación con otras propuestas heurísticas referenciadas en la literatura.

- El método RankFSP es una de las propuestas novedosas y más prometedoras de esta investigación. Este algoritmo de L2R está basado en una variante adaptada y mejo-

rada de FSP y es capaz de construir una función de *ranking* sobre un determinado conjunto de entrenamiento optimizando directamente cualquier medida de evaluación de RI. Esta variante adaptada contempla dos pilares principales: una estrategia de lanzamientos condicionados al contexto y una estrategia de reinicios de puntos de captura.

- Con el objetivo de mejorar la tarea del *ranking* se propuso también un nuevo método para la selección de rasgos. Este método simple y eficaz logra identificar los rasgos más representativos a nivel de consulta, y potenciar con ellos una sinergia en cuanto a rendimiento a nivel de colección. Esta selección contribuyó a una mejor generalización de los modelos de *ranking*.

- Los resultados de la fase de evaluación y del análisis estadístico mostraron las potencialidades de RankFSP para abordar la tarea del L2R, para los cuales siempre alcanzó los mejores valores de precisión en el *ranking* en comparación con los demás algoritmos estudiados, y en relación a las colecciones de gran tamaño MSLR-WEB10K y MSLR-WEB30K.

- Los resultados que mostraron RankFSP, RankPSO y RankSupernova frente a los conjuntos reducidos reafirmaron significativamente las ventajas de utilizar el método propuesto de selección de rasgos para lograr elevar las prestaciones de los métodos de L2R, disminuyendo adicionalmente el tiempo de entrenamiento y la cantidad de consultas a evaluar.

- Se presentó la herramienta software *Learning-to-Rank Laboratory* (L2RLab) dirigida a asistir y agilizar todo el proceso experimental para desarrollar, evaluar, comparar y analizar nuevos modelos de L2R, así como para preprocesar las colecciones de datos.

- L2RLab posibilita la reutilización de códigos y recursos implementados, ofrece una mayor rapidez (ahorro de tiempo) y confiabilidad (reducción de errores) en el proceso de experimentación. Estas condiciones, le permitirán al investigador centrarse en la tarea del L2R, sin tener que preocuparse en demasía por aspectos técnicos en el diseño y preparación de los experimentos.

Un resumen más detallado de tales resultados puede ser consultado al final de cada capítulo.

8.2. Trabajos futuros.

El desarrollo del presente trabajo a propiciado diversas cuestiones que pueden ser objeto de futuras investigaciones en el campo del L2R, las cuales serán expuestas a continuación considerando cuatro perspectivas diferentes:

1. Con relación al trabajo sobre el método RankPSO.

   - Desarrollar una estrategia multi-enjambre basada en la propuesta de RankPSO con miras a mejorar los rendimientos alcanzados sobre las colecciones de datos evaluadas u otras nuevas.
   - Proponer nuevos modelos de L2R basados en la optimización multi-objetivo con enjambre de partículas. Donde tales objetivos estarán representados por las diferentes medidas de evaluación de RI.

2. Desde el punto de vista de la metaheurística FSP.

   - Generalizar e incorporar en FSP las dos estrategias de mejoras propuestas en RankFSP, es decir, la estrategia de lanzamientos condicionados al contexto y la estrategia de reinicios de puntos de captura.
   - Analizar la calidad de solución y el tiempo de respuesta de FSP bajo diferentes condiciones y problemas de optimización.
   - En dependencia de las características y situaciones hacia donde se encaminen los actuales y futuros problemas de optimización, pudiésemos evaluar las siguientes variantes o extensiones en FSP:

     1^a. La filosofía de este primer caso, parte de permitir que en cada punto de captura se realizaran un número fijo de lanzamientos de la red de pesca, sin condicionar tales lanzamientos a mejoras de posición. Ahora, a diferencia de otras variantes, cada vez que la malla es lanzada en un determinado

punto de captura, la posición de este punto de pesca es siempre actualizada con el mejor punto obtenido en la red de pesca, sin chequear si este vector de posición es mejor que el p_i de ese punto de captura. Esta idea está basada en la intuición de que pasando a través de zonas de pesca no muy prometedoras, pudiésemos llegar a las mejores zonas de captura, las que pudiesen ser inaccesibles de otro modo (según la actuación de otra variante).

2^a. En esta segunda variante se parte por dividir la zona de pesca (espacio de búsqueda) en subregiones. A cada una de estas subregiones se envía un pescador, el cual puede aplicar como estrategia de pesca su propia variante. Esta cooperativa de pescadores comparte una memoria común donde es almacenada la mejor posición (g_{best}) lograda finalmente entre todos los pescadores. Aunque todavía el paralelismo no sea usado sistemáticamente para acelerar o mejorar la efectividad de las metaheurísticas, las aplicaciones paralelas son muy robustas y abundan en la literatura (consular en [152] un estudio al respecto). En esencia, esta variante sugiere una aplicación paralela para reforzar la robustez del método.

La selección y aplicación del FSP actual o de alguna de estas variantes mencionadas u otras que se pudiesen considerar, dependen de las características y situaciones a donde se pudiesen encaminar futuros problemas de optimización.

3. Con respecto a toda la investigación relacionada con RankFSP.

   - Conducir más experimentos con nuevas colecciones de datos con el fin de verificar las potencialidades y rendimientos de RankFSP.
   - Tratar de incorporar elementos de proactividad[1] al método de RankFSP vinculado con su disposición a la hora de acometer la tarea del *ranking*.
   - Probar la efectividad de la propuesta de selección de rasgos en nuevas y refe-

[1]Actitud en la que el sujeto u organización asume el pleno control de su conducta de modo activo, lo que implica la toma de iniciativa en el desarrollo de acciones creativas y audaces para generar mejoras.

renciadas colecciones de datos.

- Mejorar la propuesta del método RankSupernova incorporándole nuevas estrategias y/o transformaciones encaminadas a mejorar sus prestaciones para abordar la tarea del L2R.

4. En relación a la herramienta software L2RLab.

   - Incorporar al L2RLab los principales exponentes (algoritmos) del estado del arte en el área del L2R.
   - Añadir al L2RLab nuevas funcionalidades vinculadas con el preprocesamiento de las colecciones de datos, y el análisis y la comparación de modelos de *ranking*.
   - Mejorar la ayuda de la aplicación, incorporando una descripción detallada de los procedimientos y funcionalidades de cada módulo.
   - Liberar el software bajo el dominio de licencia pública GNU GPL(*GNU General Public License*).

Finalmente, también pudiésemos pensar en cómo concebir nuevos modelos de *ranking* que tomen en consideración no sólo las consultas, los documentos asociados a tales consultas y los juicios de relevancia, sino además, el contexto en el que se formulan las consultas.

Apéndice A. Especificaciones sobre la producción científica del L2R

Orden	Área del conocimiento	Artículos
1	Computer Science	525
2	Mathematics	107
3	Decision Sciences	82
4	Engineering	80
5	Business, Management and Accounting	66
6	Social Sciences	46
7	Arts and Humanities	11
8	Medicine	6
9	Biochemistry, Genetics and Molecular Biology	4
10	Health Professions	2
11	Materials Science	2
12	Neuroscience	2
13	Physics and Astronomy	2
14	Earth and Planetary Sciences	2
15	Multidisciplinary	1
16	Psychology	1
17	Agricultural and Biological Sciences	1

Tabla A-1: Tabulación de la producción científica del L2R por área del conocimiento.

Orden	Autor	Artículos
1	Li, H.	28
2	Liu, T. Y.	21
3	Wang, Y.	17
4	Liu, J.	16
5	Qin, T.	15
6	Zheng, Z.	12
7	Huang, Y.	11
8	Liu, Y.	11
9	Chang, Y.	10
10	Lin, H.	10
11	Lin, J.	10
12	Macdonald, C.	10
13	Ounis, I.	10
14	Guo, J.	9
15	Lu, M.	9
16	Pan, Y.	9
17	Tang, Y.	9
18	Usunier, N.	9
19	Xu, C.	9
20	Zha, H.	9
21	De Rijke, M.	8
22	Gallinari, P.	8
23	Lan, Y.	8
24	Lin, Y.	8
25	Metzler, D.	8
26	Xu, J.	8
27	Yan, J.	8
28	Amini, M. R.	7
29	Geng, B.	7
30	He, B.	7

Tabla A-2: Tabulación de los autores con mayor productividad científica en L2R.

Orden	País	Artículos
1	China	245
2	United States	172
3	United Kingdom	47
4	Canada	23
5	France	22
6	Germany	22
7	Netherlands	21
8	Taiwan	20
9	Italy	17
10	Singapore	16
11	Spain	15
12	Hong Kong	11
13	South Korea	7
14	India	7
15	Brazil	6
16	Japan	6
17	Israel	5
18	Australia	4
19	Portugal	4
20	Finland	4
21	Russian Federation	3
22	Switzerland	3
23	Saudi Arabia	2
24	Norway	1
25	Ireland	1
26	Hungary	1
27	Slovenia	1
28	South Africa	1
29	Greece	1
30	Cuba	1
31	Belgium	1
32	Turkey	1
33	New Zealand	1
34	Malaysia	1

Tabla A-3: Tabulación de la producción científica por países.

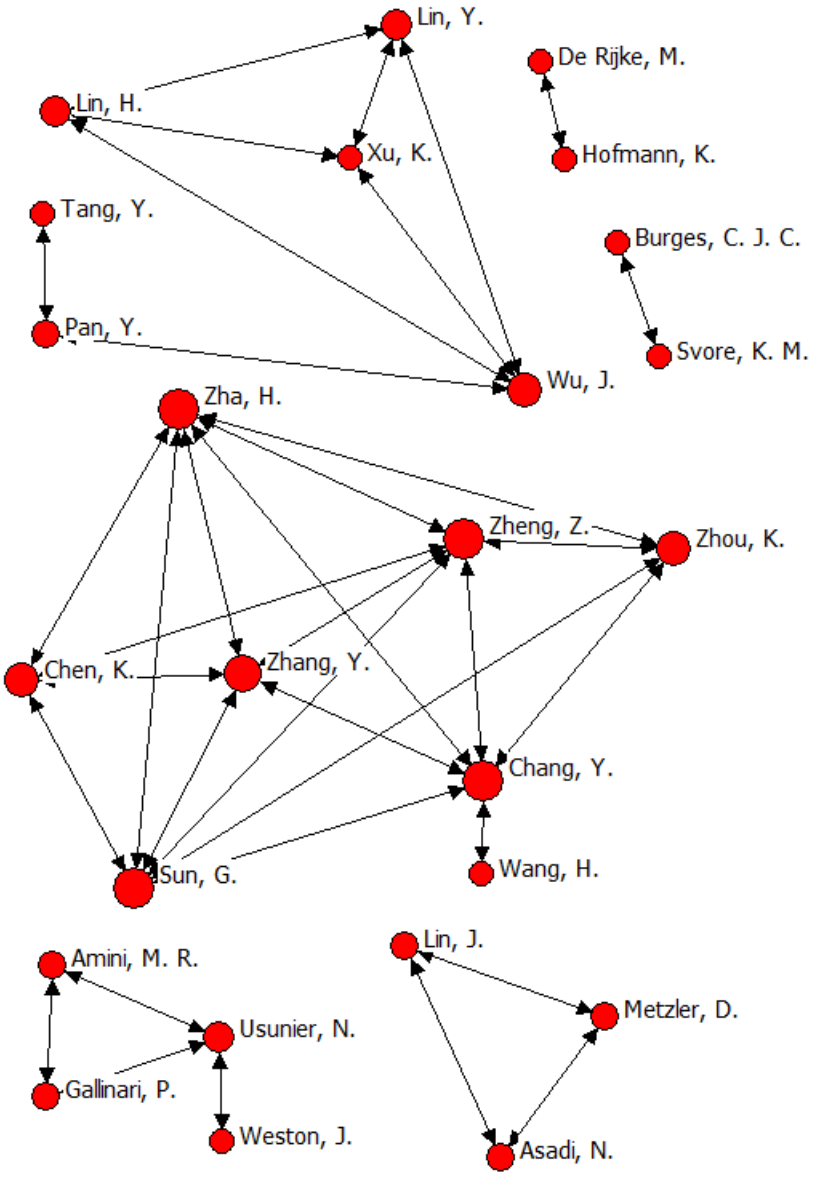


Figura A-4: Redes de colaboración entre autores de L2R.

Apéndice B. Rendimientos obtenidos por RankPSO en comparación con métodos publicados.

De los resultados alcanzados por cada uno de los métodos en TD2004 (véase Tabla B-1) se puede apreciar como *RankPSO-Div-Red* supera la precisión de todos los algoritmos en las primeras cuatro posiciones del *ranking*, sólo FRank lo iguala en la posición 3 y AdaRank-MAP en la posición 4. En las restantes posiciones, su tasa de rendimiento desciende a una condición media promedio.

Algoritmos	P@1	P@2	P@3	P@4	P@5	P@6	P@7	P@8	P@9	P@10
Regression	(-)0,36	(-)0,34	(-)0,3333	(-)0,32	(-)0,312	(-)0,2867	(-)0,2705	(=)0,2683	(+)0,2593	(-)0,2493
RankSVM	(-)0,4133	(-)0,3467	(-)0,3467	(-)0,3333	(-)0,3013	(-)0,2867	(-)0,28	(-)0,2667	(-)0,2548	(-)0,252
ListNet	(-)0,36	(-)0,3467	(-)0,36	(-)0,3367	(-)0,3067	(-)0,2933	(-)0,2857	(+)0,2717	(+)0,2637	(+)0,256
AdaRank-MAP	(-)0,4133	(-)0,3933	(-)0,3689	(=)0,3533	(-)0,328	(-)0,3111	(=)0,2914	(+)0,2733	(+)0,2637	(-)0,2493
AdaRank-NDCG	(-)0,4267	(-)0,38	(-)0,3644	(-)0,3333	(-)0,328	(-)0,3	(-)0,2838	(+)0,2783	(+)0,2622	(-)0,248
AdaRank-NDCG-Red	(-)0,427	(-)0,38	(-)0,3656	(-)0,334	(-)0,3291	(-)0,3	(-)0,284	(+)0,2781	(+)0,263	(-)0,248
SVMmap	(-)0,2933	(-)0,3067	(-)0,3022	(-)0,2933	(-)0,2907	(-)0,2756	(-)0,2743	(-)0,2617	(-)0,2519	(-)0,2467
RankPSO	(-)0,36	(-)0,38	(-)0,36	(-)0,35667	(-)0,328	(-)0,3022	(=)0,2914	(+)0,28	(+)0,2725	(+)0,2587
RankPSO-Div-Red	0,5067	0,4533	0,3778	0,3533	0,3307	0,3156	0,2914	0,2683	0,2578	0,2533
FRank	(-)0,4933	(-)0,4067	(=)0,3778	(-)0,3267	(+)0,3333	(=)0,3156	(+)0,299	(+)0,2817	(+)0,2681	(+)0,2627
Regression+L2reg	(-)0,2933	(-)0,32	(-)0,2978	(-)0,2833	(-)0,272	(-)0,2644	(-)0,2629	(-)0,25	(-)0,2415	(-)0,2347
RankSVM-Primal	(-)0,3067	(-)0,3067	(-)0,3156	(-)0,2933	(-)0,2933	(-)0,2778	(-)0,2648	(-)0,255	(-)0,2533	(-)0,2427
RankSVM-Struct	(-)0,3467	(-)0,3467	(-)0,3333	(-)0,3167	(-)0,296	(-)0,2822	(-)0,2724	(+)0,275	(+)0,2593	(+)0,256
SmoothRank	(-)0,4	(-)0,42	(-)0,3689	(-)0,3467	(-)0,32	(-)0,2956	(+)0,2933	(+)0,2883	(+)0,2785	(+)0,2653

Tabla B-1: Rendimiento de algoritmos estudiados, considerando P@1 hasta P@10 en TD2004.

En cuanto a TD2003, *RankPSO-Div-Red* supera en un 92 % los resultados alcanzados por todos los métodos de L2R en las primeras cinco posiciones del *ranking* y en un 58 % a partir de la posición 6 a la 10. Estos resultados pueden ser constatados en la siguiente Tabla B-2.

Analizando los rendimientos en NP2004 expuestos en la Tabla B-3, en todas las posiciones del *ranking*, el 82 % de los métodos tiene valores de precisión inferiores a *RankPSO-Div-Red*, sólo el 16 % lo iguala y el 2 % lo supera en las posiciones 1, 6 y 7.

En NP2003, según los valores de la Tabla B-4, *RankPSO-Div-Red* obtiene el mejor valor de precisión en la primera posición del *ranking*, sólo igualado por RankSVM, AdaRank-MAP y SmoothRank. Su rendimiento desciende a un valor promedio entre todos los algoritmos

para las posiciones 2, 3 y 4, pero a partir de la posición 5 a la 10, alcanza de nuevo la primacía (mejor posición) en el *ranking*.

Algoritmos	**P@1**	**P@2**	**P@3**	**P@4**	**P@5**	**P@6**	**P@7**	**P@8**	**P@9**	**P@10**
Regression	(-)0,32	(-)0,3	(-)0,26	(-)0,245	(-)0,216	(-)0,2133	(-)0,2057	(=)0,195	(-)0,1844	(+)0,178
RankSVM	(-)0,32	(-)0,31	(-)0,2933	(=)0,285	(+)0,276	(+)0,25	(+)0,2229	(+)0,2075	(+)0,2	(+)0,188
ListNet	(+)0,4	(-)0,33	(-)0,2933	(-)0,255	(=)0,252	(+)0,2533	(+)0,2486	(+)0,2225	(+)0,2133	(+)0,2
AdaRank-MAP	(-)0,26	(-)0,29	(-)0,26	(-)0,225	(-)0,208	(-)0,2	(-)0,1943	(-)0,18	(-)0,1667	(-)0,158
AdaRank-NDCG	(-)0,36	(-)0,27	(-)0,24	(-)0,225	(-)0,204	(-)0,1933	(-)0,1971	(-)0,1825	(-)0,1733	(-)0,164
AdaRank-NDCG-Red	(-)0,36	(-)0,28	(-)0,25	(-)0,23	(-)0,211	(-)0,194	(-)0,1972	(-)0,1831	(-)0,173	(-)0,164
SVMmap	(-)0,32	(-)0,31	(-)0,2533	(-)0,26	(-)0,232	(-)0,21	(-)0,1971	(-)0,1825	(-)0,1733	(-)0,17
RankPSO	(-)0,34	(-)0,3	(-)0,2867	(-)0,25	(-)0,224	(-)0,2067	(-)0,2	(-)0,1875	(-)0,18	(-)0,172
RankPSO-Div-Red	0,39	0,34	0,3	0,285	0,252	0,2367	0,2171	0,195	0,1867	0,174
FRank	(-)0,3	(-)0,27	(-)0,2333	(-)0,2	(-)0,172	(-)0,1733	(-)0,1657	(-)0,1575	(-)0,1533	(-)0,152
Regression+L2reg	(-)0,34	(=)0,34	(=)0,3	(-)0,255	(-)0,236	(-)0,2067	(-)0,2086	(+)0,1975	(-)0,1822	(+)0,176
RankSVM-Primal	(-)0,32	(+)0,35	(-)0,2933	(=)0,285	(+)0,272	(+)0,2533	(+)0,2314	(+)0,21	(+)0,2044	(+)0,194
RankSVM-Struct	(-)0,34	(-)0,32	(-)0,2867	(=)0,285	(+)0,276	(+)0,25	(+)0,2343	(+)0,2075	(+)0,2	(+)0,186
SmoothRank	(-)0,38	(=)0,34	(-)0,2733	(-)0,26	(-)0,244	(-)0,2267	(+)0,2229	(+)0,21	(+)0,1956	(+)0,188

Tabla B-2: Rendimiento de algoritmos estudiados, considerando P@1 hasta P@10 en TD2003.

Algoritmos	**P@1**	**P@2**	**P@3**	**P@4**	**P@5**	**P@6**	**P@7**	**P@8**	**P@9**	**P@10**
Regression	(-)0,3733	(-)0,2667	(-)0,2	(-)0,1733	(-)0,144	(-)0,1311	(-)0,1181	(-)0,1033	(-)0,0919	(-)0,0827
RankSVM	(-)0,5067	(=)0,3733	(-)0,2622	(=)0,22	(-)0,1787	(-)0,1489	(-)0,1295	(-)0,1133	(-)0,1022	(-)0,0933
ListNet	(-)0,5333	(=)0,3733	(-)0,2667	(-)0,2167	(-)0,1787	(=)0,1533	(-)0,1333	(-)0,1183	(-)0,1052	(-)0,0947
AdaRank-MAP	(-)0,48	(-)0,3333	(-)0,2444	(-)0,1933	(-)0,1627	(-)0,14	(-)0,1219	(-)0,11	(-)0,0978	(-)0,088
AdaRank-NDCG	(-)0,5067	(-)0,32	(-)0,2489	(-)0,2067	(-)0,1653	(-)0,1422	(-)0,1238	(-)0,1133	(-)0,1007	(-)0,0907
AdaRank-NDCG-Red	(-)0,5067	(-)0,321	(-)0,249	(-)0,2067	(-)0,1654	(-)0,142	(-)0,1235	(-)0,113	(-)0,101	(-)0,0909
SVMmap	(-)0,52	(-)0,36	(-)0,2667	(=)0,22	(-)0,1787	(=)0,1533	(-)0,1333	(-)0,1183	(-)0,1052	(=)0,096
RankPSO	(-)0,5067	(-)0,367	(-)0,2667	(-)0,2133	(=)0,1813	(=)0,1533	(-)0,1333	(-)0,1183	(=)0,1067	(=)0,096
RankPSO-Div-Red	0,56	0,3733	0,2756	0,22	0,1813	0,1533	0,1352	0,12	0,1067	0,096
FRank	(-)0,48	(-)0,3133	(-)0,2356	(-)0,19	(-)0,16	(-)0,14	(-)0,1238	(-)0,11	(-)0,1007	(-)0,0933
Regression+L2reg	(+)0,5733	(-)0,36	(-)0,2622	(-)0,2167	(-)0,176	(=)0,1533	(=)0,1352	(-)0,1183	(=)0,1067	(=)0,096
RankSVM-Primal	(=)0,56	(-)0,36	(-)0,2533	(-)0,2133	(-)0,1733	(-)0,1511	(-)0,1314	(-)0,1167	(-)0,1052	(-)0,0947
RankSVM-Struct	(=)0,56	(-)0,36	(-)0,2578	(-)0,2167	(-)0,1733	(-)0,1511	(-)0,1314	(-)0,1167	(-)0,1052	(-)0,0947
SmoothRank	(-)0,5467	(-)0,36	(-)0,2667	(=)0,22	(-)0,1787	(+)0,1556	(+)0,1371	(=)0,12	(=)0,1067	(=)0,096

Tabla B-3: Rendimiento de algoritmos estudiados, considerando P@1 hasta P@10 en NP2004.

Algoritmos	**P@1**	**P@2**	**P@3**	**P@4**	**P@5**	**P@6**	**P@7**	**P@8**	**P@9**	**P@10**
Regression	(-)0,4467	(-)0,2833	(-)0,22	(-)0,1767	(-)0,1453	(-)0,1256	(-)0,1105	(-)0,0983	(-)0,0889	(-)0,0813
RankSVM	(=)0,58	(+)0,37	(+)0,2711	(-)0,2083	(-)0,1707	(-)0,1433	(=)0,1267	(-)0,1117	(-)0,1	(-)0,092
ListNet	(-)0,5667	(+)0,37	(+)0,2667	(-)0,2083	(-)0,172	(-)0,1456	(+)0,1295	(=)0,1133	(-)0,1015	(=)0,0927
AdaRank-MAP	(=)0,58	(-)0,3567	(-)0,2511	(-)0,1917	(-)0,16	(-)0,1367	(-)0,1181	(-)0,105	(-)0,0941	(-)0,0867
AdaRank-NDCG	(-)0,56	(-)0,3467	(-)0,2467	(-)0,1967	(-)0,1613	(-)0,14	(-)0,1229	(-)0,1083	(-)0,0985	(-)0,0893
AdaRank-NDCG-Red	(-)0,565	(-)0,3469	(-)0,2467	(-)0,197	(-)0,1617	(-)0,141	(-)0,1231	(-)0,108	(-)0,0986	(-)0,0892
SVMmap	(-)0,56	(+)0,38	(+)0,2689	(=)0,21	(-)0,1707	(-)0,1433	(-)0,1257	(-)0,11	(-)0,0985	(-)0,0893
RankPSO	(=)0,58	(-)0,3467	(-)0,2488	(-)0,202	(-)0,1666	(-)0,1421	(-)0,1211	(-)0,1086	(-)0,1	(-)0,0925
RankPSO-Div-Red	0,58	0,3633	0,2644	0,21	0,1747	0,1467	0,1267	0,1133	0,103	0,0927
FRank	(-)0,54	(-)0,3533	(-)0,2533	(-)0,2033	(-)0,168	(-)0,1433	(-)0,1238	(-)0,11	(-)0,0978	(-)0,0907
Regression+L2reg	(-)0,5467	(+)0,37	(+)0,2733	(+)0,2117	(-)0,172	(-)0,1456	(=)0,1267	(-)0,1117	(-)0,1	(-)0,0913
RankSVM-Primal	(-)0,5733	(-)0,36	(+)0,2733	(+)0,2117	(-)0,1693	(-)0,1422	(-)0,1257	(-)0,11	(-)0,0993	(-)0,0907
RankSVM-Struct	(-)0,5533	(+)0,3667	(-)0,2622	(-)0,2083	(-)0,1707	(-)0,1433	(-)0,1257	(-)0,1117	(-)0,1	(-)0,092
SmoothRank	(=)0,58	(+)0,3733	(-)0,26	(-)0,2083	(-)0,172	(-)0,1456	(-)0,1257	(-)0,11	(-)0,1	(-)0,09

Tabla B-4: Rendimiento de algoritmos estudiados, considerando P@1 hasta P@10 en NP2003.

La Tabla B-5 muestra como nuestro método obtiene el mejor valor de precisión en HP2004 para la primera posición del *ranking*, con una diferencia promedio de 0.06 unidades en relación al valor de los demás métodos. A partir de las posiciones 2 a la 10, este se mantiene como promedio en la cuarta posición del *ranking*.

Algoritmos	**P@1**	**P@2**	**P@3**	**P@4**	**P@5**	**P@6**	**P@7**	**P@8**	**P@9**	**P@10**
Regression	(-)0,3867	(-)0,2867	(-)0,2133	(-)0,1767	(-)0,144	(-)0,1267	(-)0,1086	(-)0,0983	(-)0,0919	(-)0,084
RankSVM	(-)0,5733	(-)0,3667	(-)0,2667	(=)0,2167	(-)0,1787	(-)0,1533	(-)0,1314	(-)0,1167	(-)0,1052	(-)0,096
ListNet	(-)0,6	(=)0,3733	(-)0,2711	(+)0,23	(+)0,1893	(+)0,1622	(+)0,141	(+)0,1233	(-)0,1096	(-)0,0987
AdaRank-MAP	(-)0,6133	(+)0,4067	(+)0,2978	(+)0,23	(+)0,1867	(+)0,1578	(-)0,1352	(-)0,1183	(-)0,1052	(-)0,0947
AdaRank-NDCG	(-)0,5867	(+)0,3867	(-)0,2711	(+)0,2233	(=)0,1813	(-)0,1533	(-)0,1314	(-)0,1167	(-)0,1052	(-)0,096
AdaRank-NDCG-Red	(-)0,59	(+)0,38	(-)0,2732	(+)0,23	(-)0,1811	(-)0,1513	(-)0,1309	(-)0,1154	(-)0,1051	(-)0,097
SVMmap	(-)0,6267	(+)0,4	(-)0,28	(+)0,2333	(+)0,1893	(+)0,16	(=)0,1371	(=)0,12	(-)0,1067	(-)0,096
RankPSO	(-)0,59	(-)0,37	(-)0,269	(+)0,228	(+)0,1831	(+)0,1589	(+)0,1398	(=)0,12	(-)0,106	(-)0,097
RankPSO-Div-Red	0,63	0,3733	0,2844	0,2167	0,1813	0,1556	0,1371	0,12	0,1111	0,1013
FRank	(-)0,6	(=)0,3733	(-)0,2622	(-)0,2033	(-)0,168	(-)0,1444	(-)0,1238	(-)0,1083	(-)0,0978	(-)0,0893
Regression+L2reg	(-)0,5333	(-)0,34	(-)0,2489	(-)0,2033	(-)0,168	(-)0,1422	(-)0,1257	(-)0,11	(-)0,1007	(-)0,0907
RankSVM-Primal	(-)0,5733	(=)0,3733	(-)0,2667	(=)0,2167	(-)0,1787	(=)0,1556	(-)0,1352	(-)0,1183	(-)0,1067	(-)0,096
RankSVM-Struct	(-)0,5867	(-)0,3667	(-)0,2756	(+)0,22	(-)0,1787	(=)0,1556	(-)0,1333	(-)0,1183	(-)0,1052	(-)0,0947
SmoothRank	(-)0,6133	(+)0,4	(+)0,2978	(+)0,23	(+)0,1947	(+)0,1644	(+)0,141	(+)0,1233	(-)0,1096	(-)0,0987

Tabla B-5: Rendimiento de algoritmos estudiados, considerando P@1 hasta P@10 en HP2004.

Finalmente para el caso de HP2003, *RankPSO-Div-Red* presenta un comportamiento y rendimiento muy similar al mostrado en HP2004 en comparación con los demás métodos de L2R. Estos valores de precisión se presentan en la siguiente Tabla B-6.

Algoritmos	**P@1**	**P@2**	**P@3**	**P@4**	**P@5**	**P@6**	**P@7**	**P@8**	**P@9**	**P@10**
Regression	(-)0,42	(-)0,27	(-)0,2111	(-)0,1733	(-)0,1453	(-)0,1233	(-)0,1143	(-)0,1008	(-)0,0926	(-)0,0887
RankSVM	(-)0,6933	(-)0,4233	(-)0,3089	(=)0,24	(=)0,1987	(-)0,1689	(-)0,1467	(-)0,1292	(-)0,1148	(-)0,104
ListNet	(-)0,72	(+)0,4533	(+)0,32	(+)0,2483	(+)0,204	(+)0,1722	(+)0,1495	(=)0,1308	(=)0,117	(+)0,1067
AdaRank-MAP	(-)0,7333	(+)0,45	(-)0,3089	(=)0,24	(=)0,1987	(-)0,1689	(-)0,1476	(-)0,13	(-)0,1163	(+)0,106
AdaRank-NDCG	(-)0,7133	(-)0,4367	(=)0,3111	(-)0,2367	(-)0,1947	(-)0,1644	(-)0,1419	(-)0,1242	(-)0,1104	(-)0,1
AdaRank-NDCG-Red	(-)0,7145	(-)0,437	(+)0,3112	(-)0,2367	(-)0,1948	(-)0,1645	(-)0,142	(-)0,1242	(-)0,1104	(-)0,1
SVMmap	(-)0,7133	(-)0,4333	(-)0,3089	(=)0,24	(-)0,1947	(-)0,1656	(-)0,1429	(-)0,125	(-)0,1111	(-)0,1
RankPSO	(-)0,7178	(-)0,439	(+)0,3145	(-)0,239	(-)0,1947	(-)0,1681	(-)0,1419	(-)0,126	(-)0,1132	(-)0,102
RankPSO-Div-Red	0,74	0,44	0,3111	0,24	0,1987	0,1711	0,1486	0,1308	0,117	0,1053
FRank	(-)0,6533	(-)0,4133	(-)0,2889	(-)0,2333	(=)0,1987	(-)0,17	(-)0,1467	(-)0,1283	(-)0,1156	(+)0,1067
Regression+L2reg	(-)0,6933	(+)0,4467	(+)0,32	(+)0,2467	(-)0,1973	(-)0,1656	(-)0,1419	(-)0,125	(-)0,1111	(-)0,1027
RankSVM-Primal	(=)0,74	(-)0,4333	(+)0,3133	(+)0,2433	(+)0,2	(-)0,1689	(-)0,1467	(-)0,1283	(-)0,1148	(-)0,104
RankSVM-Struct	(=)0,74	(-)0,4333	(-)0,3089	(=)0,24	(=)0,1987	(-)0,1667	(-)0,1438	(-)0,1283	(-)0,1141	(-)0,1027
SmoothRank	(-)0,7133	(+)0,4467	(+)0,3244	(+)0,2517	(+)0,2067	(+)0,1733	(=)0,1486	(-)0,13	(-)0,1156	(=)0,1053

Tabla B-6: Rendimiento de algoritmos estudiados, considerando P@1 hasta P@10 en HP2003.

Apéndice C. Resultados del estudio estadístico en colecciones estándares de prueba.

N	150
Chi-Square	165,324
df	9
Asymp. Sig.	,000

Tabla C-1: Estadísticos obtenidos por la prueba de Friedman en HP2003.

Algoritmos	**Rango promedio**
AdaRankMAP	5,86
AdaRankNDCG	5,68
AdaRankNDCG-Red	5,68
FRank	5,41
ListNet	5,91
RankBoost	5,60
RankPSO	5,77
RankPSO-Div-Red	5,95
RankSVM	5,59
Regression	3,56

Tabla C-2: Rangos en HP2003.

N	150
Chi-Square	64,243
df	9
Asymp. Sig.	,000

Tabla C-3: Estadísticos obtenidos por la prueba de Friedman en HP2004.

Algoritmos	**Rango promedio**
AdaRankMAP	5,89
AdaRankNDCG	5,44
AdaRankNDCG-Red	5,44
FRank	5,45
ListNet	5,89
RankBoost	5,24
RankPSO	5,60
RankPSO-Div-Red	5,93
RankSVM	5,85
Regression	4,28

Tabla C-4: Rangos en HP2004.

N	150
Chi-Square	76,526
df	9
Asymp. Sig.	,000

Tabla C-5: Estadísticos obtenidos por la prueba de Friedman en NP2003.

Algoritmos	**Rango promedio**
AdaRankMAP	5,41
AdaRankNDCG	5,49
AdaRankNDCG-Red	5,49
FRank	5,45
ListNet	5,97
RankBoost	6,09
RankPSO	5,50
RankPSO-Div-Red	5,45
RankSVM	5,90
Regression	4,26

Tabla C-6: Rangos en NP2003.

N	75
Chi-Square	44,991
df	9
Asymp. Sig.	,000

Tabla C-7: Estadísticos obtenidos por la prueba de Friedman en NP2004.

Algoritmos	**Rango promedio**
AdaRankMAP	5,39
AdaRankNDCG	5,63
AdaRankNDCG-Red	5,67
FRank	5,47
ListNet	5,71
RankBoost	4,28
RankPSO	6,03
RankPSO-Div-Red	6,18
RankSVM	5,96
Regression	4,68

Tabla C-8: Rangos en NP2004.

N	50
Chi-Square	41,764
df	9
Asymp. Sig.	,000

Tabla C-9: Estadísticos obtenidos por la prueba de Friedman en TD2003.

N	75
Chi-Square	54,894
df	9
Asymp. Sig.	,000

Tabla C-10: Estadísticos obtenidos por la prueba de Friedman en TD2004.

Algoritmos	**Rango promedio**
AdaRankMAP	4,76
AdaRankNDCG	4,67
AdaRankNDCG-Red	4,87
FRank	4,67
ListNet	7,08
RankBoost	5,17
RankPSO	5,48
RankPSO-Div-Red	6,96
RankSVM	5,98
Regression	5,36

Tabla C-11: Rangos en TD2003.

Algoritmos	**Rango promedio**
AdaRankMAP	4,93
AdaRankNDCG	4,41
AdaRankNDCG-Red	4,55
FRank	6,23
ListNet	5,18
RankBoost	7,33
RankPSO	5,29
RankPSO-Div-Red	5,82
RankSVM	5,76
Regression	5,49

Tabla C-12: Rangos en TD2004.

Las comparaciones realizadas entre los algoritmos de L2R, siguiendo el *test* de Wilcoxon revelan que en HP2003 *RankPSO-Div-Red* mejora significativamente los rendimientos de los métodos RankBoost y RankSVM, y supera de manera altamente significativa a FRank. Los métodos AdaRank-MAP, AdaRank-NDCG y *AdaRank-NDCG-Red* presentan mejoras medianamente significativas frente a FRank. Por su parte, ListNet también supera con un alto nivel de significación el rendimiento de FRank y de forma significativa a RankSVM.

En el conjunto de datos HP2004, *RankPSO-Div-Red*, *RankPSO*, RankSVM, ListNet y AdaRank-MAP superan de manera altamente significativa la precisión alcanzada por

RankBoost. Además, *RankPSO-Div-Red* y RankSVM superan moderadamente el rendimiento de los métodos AdaRank-NDCG, *AdaRank-NDCG-Red* y FRank.

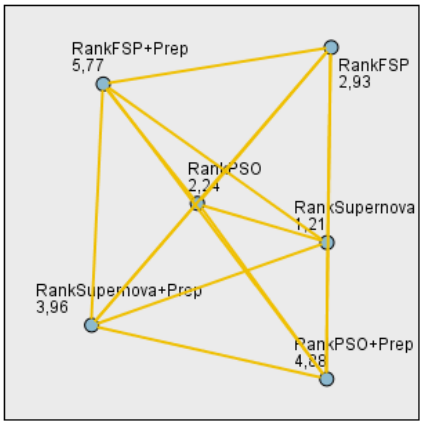
En NP2003, los métodos AdaRank-NDCG, *AdaRank-NDCG-Red* y FRank tienen un comportamiento medianamente inferior a RankBoost y RankSVM. En este *dataset*, *RankPSO* alcanza un rendimiento significativamente superior a AdaRank-MAP.

En el conjunto de datos TD2003, es perceptible que el comportamiento del rendimiento de *RankPSO-Div-Red* y ListNet superan con un alto nivel de significación a todos los métodos de L2R. *RankPSO* y Regression presentan mejoras moderamente significativas en relación a FRank, mientras que la precisión alcanzada por RankSVM supera con gran significación a FRank. Por otra parte, AdaRank-NDCG es superado significativamente por el método *AdaRank-NDCG-Red*.

En TD2004, y en particular, las medias de precisión alcanzadas por AdaRank-NDCG son inferiores estadísticamente a todos los demás métodos de L2R (excepto Regression). *RankPSO-Div-Red* mejora con alto nivel de significación los rendimientos de AdaRank-NDCG y *AdaRank-NDCG-Red*, y supera significativamente a ListNet. FRank de manera ventajosa también tiene, en cuanto a precisión, grandes diferencias significativas con respecto a AdaRank-NDCG y *AdaRank-NDCG-Red*, moderadamente sobre Regression y significativas comparado con AdaRank-MAP. En este *dataset*, y en contraposición al desempeño mostrado en NP2004, RankBoost es capaz de superar estadísticamente el rendimiento de todos los demás métodos de aprendizaje. AdaRank-MAP, ListNet y RankSVM mejoran moderadamente la media de precisión de *AdaRank-NDCG-Red*; mientras RankSVM supera significativamente a AdaRank-MAP, y RankPSO de igual manera a ListNet.

Apéndice D. Resultados del estudio estadístico en la colección MSLR-WEB30K.

(a) Gráfico de distancias de red



(b) Tabla de comparaciones

Muestra1-Muestra2	Prueba estadística	Error típico	Desv. Prueba estadística	Sig.	Sig. ady.
RankSupernova-RankPSO	-1,036	,015	-69,528	,000	,000
RankSupernova-RankFSP	-1,719	,015	-115,362	,000	,000
RankSupernova-RankSupernova+Prep	-2,755	,015	-184,889	,000	,000
RankSupernova-RankPSO+Prep	-3,673	,015	-246,507	,000	,000
RankSupernova-RankFSP+Prep	-4,563	,015	-306,264	,000	,000
RankPSO-RankFSP	-,683	,015	-45,834	,000	,000
RankPSO-RankSupernova+Prep	-1,719	,015	-115,362	,000	,000
RankPSO-RankPSO+Prep	-2,637	,015	-176,980	,000	,000
RankPSO-RankFSP+Prep	-3,527	,015	-236,736	,000	,000
RankFSP-RankSupernova+Prep	-1,036	,015	-69,528	,000	,000
RankFSP-RankPSO+Prep	-1,954	,015	-131,146	,000	,000
RankFSP-RankFSP+Prep	-2,844	,015	-190,902	,000	,000
RankSupernova+Prep-RankPSO+Prep	-,918	,015	61,618	,000	,000
RankSupernova+Prep-RankFSP+Prep	-1,808	,015	121,375	,000	,000
RankPSO+Prep-RankFSP+Prep	-,890	,015	-59,757	,000	,000

Figura D-1: Resumen de las comparaciones múltiples por parejas de los algoritmos en MSLR-WEB30K.

Bibliografía

[1] T.-Y. Liu, "Learning to rank for information retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[2] H. Li, *Learning to Rank for Information Retrieval and Natural Language Processing*. 2011.

[3] T. Joachims, "A support vector method for multivariate performance measures," in *Proceedings of the 22nd Annual International Conference on Machine Learning*, (New York, USA), pp. 377–384, 2005.

[4] Y. Jen-Yuan, L. Yung-Yi, K. Hao-Ren, and Y. Wei-Pang, "Learning to rank for information retrieval using genetic programming," in *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, (Amsterdam, Netherlands.), pp. 23–27, 2007.

[5] J. Xu, T.-Y. Liu, M. Lu, H. Li, and W.-Y. Ma, "Directly optimizing evaluation measures in learning to rank," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 107–114, 2008.

[6] H. Valizadegan, R. Jin, R. Zhang, and J. Mao, "Learning to rank by optimizing ndcg measure," in *Proceedings of 23rd Annual Conference on Neural Information Processing Systems*, pp. 1883–1891, 2009.

[7] J. Wu, Z. Yang, Y. Lin, H. Lin, Z. Ye, and K. Xu, "Learning to rank using query-level regression," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, (Beijing, China), pp. 1091–1092, 2011.

[8] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon, "Adapting ranking svm to document retrieval," in *Proceedings of the 29th annual international ACM SIGIR*

conference on Research and development in information retrieval, (New York, USA), pp. 186–193, ACM, 2006.

[9] J. Xu and H. Li, "AdaRank: a boosting algorithm for information retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 391–398, 2007.

[10] Y. Lin, H. Lin, K. Xu, and X. Sun, "Learning to rank using smoothing methods for language modeling," *Journal of the American Society for Information Science and Technology*, vol. 64, no. 4, pp. 818–828, 2013.

[11] R. Nallapati, "Discriminative models for information retrieval," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 64–71, 2004.

[12] P. Li, C. J. Burges, and Q. Wu, "Mcrank: Learning to Rank Using Multiple Classification and Gradient Boosting," in *Advances in Neural Information Processing Systems 20*, (Cambridge, MA), pp. 897–904, 2008.

[13] D. Cossock and T. Zhang, "Subset ranking using regression," in *Proceedings of the 19th annual conference on Learning Theory*, pp. 605–619, 2006.

[14] K. Crammer and Y. Singer, "Pranking with Ranking," in *Proceedings of the 15th Annual Conference on Neural Information Processing Systems*, pp. 641–647, 2001.

[15] W. Chu and S. S. Keerthi, "New approaches to support vector ordinal regression," in *Proceedings of the 22nd international conference on Machine learning*, (New York, USA), pp. 145–152, 2005.

[16] T. Joachims, "Optimizing Search Engines Using Clickthrough Data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, USA), pp. 133–142, 2002.

[17] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *The Journal of Machine Learning Research*, vol. 4, pp. 933–969, 2003.

[18] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to Rank using Gradient Descent," in *ICML '05 Proceedings of the 22nd international conference on Machine learning*, (New York, USA), pp. 89–96, ACM, 2005.

[19] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma, "Frank: a ranking method with fidelity loss," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 383–390, 2007.

[20] T. Qin, X.-D. Zhang, D.-S. Wang, T.-Y. Liu, W. Lai, and H. Li, "Ranking with Multiple Hyperplanes," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 279–286, 2007.

[21] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 271–278, 2007.

[22] M. Taylor, J. Guiver, S. Robertson, and T. Minka, "SoftRank: optimizing non-smooth rank metrics," in *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pp. 77–86, 2008.

[23] Y. Yue and C. J. C. Burges, "On using simultaneous perturbation stochastic approximation for IR measures, and the empirical optimality of lambdarank," in *Proceedings of NIPS Machine Learning for Web Search Workshop*, (Whistler, Canada), 2007.

[24] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li, "Listwise approach to learning to rank: Theory and algorithm," in *Proceedings of the 25th international conference on Machine learning*, (New York, USA), pp. 1192–1199, 2008.

[25] C. van Rijsbergen, *Information Retrieval*. Butterworth-Heinemann, 1979.

[26] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Journal Information Processing and Management*, vol. 24, no. 5, pp. 513–523, 1988.

[27] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Magazine Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[28] S. K. M. Wong, W. Ziarko, and P. C. N. Wong, "Generalized vector spaces model in information retrieval," in *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 18–25, 1985.

[29] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by Latent Semantic Analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[30] W. B. Croft and D. J. Harper, "Using probabilistic models of document retrieval without relevance information," *Journal of Documentation*, vol. 35, no. 4, pp. 285–295, 1979.

[31] S. E. Robertson, "The probability ranking principle in IR," *Journal of Documentation*, vol. 33, no. 4, pp. 294–304, 1977.

[32] S. E. Robertson and K. S. Jones, "Relevance weighting of search terms," *Journal of the American Society for Information Science*, vol. 27, no. 3, pp. 129–146, 1976.

[33] D. D. Lewis and K. S. Jones, "Natural language processing for information retrieval," *Magazine Communications of the ACM*, vol. 39, no. 1, pp. 92–101, 1996.

[34] J. M. Ponte and W. B. Croft, "A language modeling approach to information retrieval," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 275–281, 1998.

[35] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to Ad Hoc information retrieval," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 334–342, 2001.

[36] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. 1999.

[37] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?," in *Proceedings of the IEEE*, pp. 1270–1278, 2000.

[38] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Journal Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[39] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with trustrank," in *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, pp. 576–587, 2004.

[40] Y. Liu, B. Gao, T.-Y. Liu, Y. Zhang, Z. Ma, S. H. Peking, and H. Li, "BrowseRank: letting web users vote for page importance," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 451–458, 2008.

[41] R. F. Serfozo, "Technical Note–An Equivalence Between Continuous and Discrete Time Markov Decision Processes," *Operations Research*, vol. 27, no. 3, pp. 616–620, 1979.

[42] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3," in *Third Text REtrieval Conference (TREC-3)*, pp. 109–126, 1995.

[43] T. Qin, T. Y. Liu, J. Xu, and H. Li, "LETOR: A benchmark collection for research on learning to rank for information retrieval," *Information Retrieval*, vol. 13, no. 4, pp. 346–374, 2010.

[44] K. Järvelin and J. Kekäläinen, "Cumulated Gain-Based Evaluation of IR Techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[45] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan, "Expected reciprocal rank for graded relevance," in *Proceedings of the 18th ACM conference on Information and knowledge management*, (New York, USA), pp. 621–630, 2009.

[46] A. Moffat and J. Zobel, "Rank-biased precision for measurement of retrieval effectiveness," *ACM Transactions on Information Systems (TOIS)*, vol. 27, no. 1, pp. 1–27, 2008.

[47] E. Voorhees, "Overview of the trec 2002 question answering track," in *Proceedings of the Eleventh Text Retrieval Conference(TREC)*, pp. 115–123, 2002.

[48] X. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, and H.-Y. Shum, "Query dependent ranking using k-nearest neighbor," in *Proceedings of the 31st annual international ACMSIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 115–122, 2008.

[49] I. Matveeva, C. Burges, T. Burkard, A. Laucius, and L. Wong, "High accuracy retrieval with multiple nested ranker," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 437–444, 2006.

[50] N. Fuhr, "Optimum polynomial retrieval functions based on the probability ranking principle," *ACM Transactions on Information Systems (TOIS)*, vol. 7, no. 3, pp. 183–204, 1989.

[51] D. Cossock and T. Zhang, "Subset ranking using regression," in *Proceedings of the 19th annual conference on Learning Theory*, pp. 605–619, 2006.

[52] S. Kramer, G. Widmer, B. Pfahringer, and M. D. Groeve, "Prediction of Ordinal Classes Using Regression Trees," *Journal Fundamenta Informaticae - Intelligent Systems*, vol. 47, no. 1-2, pp. 1–13, 2001.

[53] R. Nallapati, "Discriminative models for information retrieval," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 64–71, 2004.

[54] P. Li, C. J. C. Burges, and Q. Wu, "McRank: Learning to Rank Using Multiple Classification and Gradient Boosting," in *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems*, 2007.

[55] W. Chu and Z. Ghahramani, "Gaussian Processes for Ordinal Regression," *The Journal of Machine Learning Research*, vol. 6, pp. 1019–1041, 2005.

[56] R. Caruana, S. Baluja, and T. Mitchell, "Using the Future to "Sort Out"the Present: Rankprop and Multitask Learning for Medical Risk Evaluation," in *In Advances in Neural Information Processing Systems 8*, pp. 959–965, 1996.

[57] K. Crammer and Y. Singer, "Pranking with Ranking," in *Advances in Neural Information Processing Systems 14*, pp. 641–647, 2001.

[58] W. Chu and S. S. Keerthi, "New approaches to support vector ordinal regression," in *Proceedings of the 22nd international conference on Machine learning*, (New York, USA), pp. 145–152, 2005.

[59] R. Herbrich, T. Graepel, and K. Obermayer, *Large margin rank boundaries for ordinal regression*, ch. 7, pp. 115–132. MIT Press, 2000.

[60] W. W. Cohen, R. E. Schapire, and Y. Singer, "Learning to order things," *Journal of Artificial Intelligence Research*, vol. 10, no. 1, pp. 243–270, 1999.

[61] P. Long and R. Servedio, "Boosting the Area under the ROC Curve.," in *Proceedings of 20th Conference on Neural Information Processing Systems (NIPS)*, p. 8, 2007.

[62] R. Herbrich, T. Graepel, and K. Obermayer, "Support vector learning for ordinal regression," in *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, pp. 97–102, 1999.

[63] J. Gao, H. Qi, X. Xia, and J.-Y. Nie, "Linear discriminant model for information retrieval," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 290–297, 2005.

[64] Z. Zheng, K. Chen, G. Sun, and H. Zha, "A regression framework for learning ranking functions using relative relevance judgments," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 287–294, 2007.

[65] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun, "A General Boosting Method and its Application to Learning Ranking Functions for Web Search," in *Advances in Neural Information Processing Systems 20* (J. Platt, D. Koller, Y. Singer, and S. Roweis, eds.), pp. 1697–1704, MIT Press, 2008.

[66] C. Cortes, M. Mohri, and A. Rastogi, "Magnitude-preserving ranking algorithms," in *Proceedings of the 24th international conference on Machine learning*, (New York, USA), pp. 169–176, 2007.

[67] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Proceedings of the 24th international conference on Machine learning*, (New York, USA), pp. 129–136, ACM, 2007.

[68] T. Qin, X.-D. Zhang, M.-F. Tsai, D.-S. Wang, T.-Y. Liu, and H. Li, "Query-level loss functions for information retrieval," *Information Processing and Management: an International Journal*, vol. 44, no. 2, pp. 838–855, 2008.

[69] M. Volkovs and R. S. Zemel, "BoltzRank: learning to maximize expected ranking gain," in *ICML, volume 382 of ACM International Conference Proceeding Series*, p. 137, 2009.

[70] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large Margin Methods for Structured and Interdependent Output Variables," *The Journal of Machine Learning Research*, vol. 6, no. 1, pp. 1453–1484, 2005.

[71] C. J. C. Burges, R. Ragno, and Q. V. Le, "Learning to rank with nonsmooth cost functions," in *Advances in Neural Information Processing Systems 19*, vol. 18, (Cambridge, MA), pp. 193–200, MIT Press, 2007.

[72] D. A. Torres, D. Turnbull, B. K. Sriperumbudur, L. Barrington, and G. R. G. Lanckriet, "Finding Musically Meaningful Words by Sparse CCA," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.

[73] D. Metzler and W. B. Croft, "A Markov random field model for term dependencies," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 472–479, 2005.

[74] O. Chapelle, Q. Le, and A. Smola, "Large margin optimization of ranking measures," in *In NIPS workshop on Machine Learning for Web Search*, p. 8, 2007.

[75] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[76] O. Chapelle, "Direct Optimization for Web Search Ranking," in *Proceedings of SIGIR 2009 Workshop on Learning to Rank for Information Retrieval*, 2009.

[77] D. A. Metzler, W. B. Croft, and A. McCallum, "Direct maximization of rank-based metrics for information retrieval," 2005.

[78] H. M. de Almeida, M. A. Gonçalves, M. Cristo, and P. Calado, "A combined component approach for finding collection-adapted ranking functions based on genetic programming," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 399–406, 2007.

[79] W. Fan, P. Pathak, and L. Wallace, "Nonlinear ranking function representations in genetic programming-based ranking discovery for personalized search," *Decision Support Systems*, vol. 42, no. 3, pp. 1338–1349, 2006.

[80] J. Guiver and E. Snelson, "Learning to rank with softrank and gaussian processes," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 259–266, 2008.

[81] S. Gorbea, "Modelación matemática de la actividad bibliotecaria: una revisión," *Investigación Bibliotecológica: archivonomía, bibliotecología e información*, vol. 12, no. 24, pp. 5–23, 1998.

[82] A. Pritchard, "Statistical bibliography or Bibliometrics," *Journal of Documentation*, vol. 25, no. 4, pp. 348–349, 1969.

[83] V. Larivière, C. R. Sugimoto, and B. Cronin, "A bibliometric chronicling of library and information science's first hundred years," *Journal of the American Society for Information Science and Technology*, vol. 63, no. 5, pp. 997–1016, 2012.

[84] M. Morales-Morejón, "La Bibliotecología, la Cienciología y la Ciencia de la Información y sus disciplinas instrumentales: Su alcance conceptual," *Ciencias de la Información*, vol. 26, no. 2, pp. 70–88, 1995.

[85] M. Morales-Morejón, "La informetría y las fuentes de información personales e institucionales. Su importancia en relación con la información de inteligencia," *Ciencias de la Información*, vol. 28, no. 3, pp. 207–217, 1997.

[86] Y. Piedra, L. Benítez, H. Saladrigas, and A. Martínez, "Análisis métrico de la producción científica en comunicación social en Cuba," *ACIMED*, vol. 14, no. 4, 2006.

[87] E. Jiménez-Contreras, *Los métodos bibliométricos. Estado de la cuestión y aplicaciones. Estudios Métricos de la Información.* La Habana: Félix Varela, 2004.

[88] J. A. Araújo and R. J. Arencibia, "Informetría, bibliometría y cienciometría : aspectos teórico-prácticos," *ACIMED*, vol. 10, no. 4, p. 4, 2002.

[89] R. Cañedo, R. Rodríguez, and M. Montejo, "Scopus: la mayor base de datos de literatura científica arbitrada al alcance de los países subdesarrollados," *ACIMED*, vol. 21, no. 3, pp. 270–282, 2010.

[90] D. Torres-Salinas and E. Jiménez-Contreras, "Introducción y estudio comparativo de los nuevos indicadores de citación sobre revistas científicas en Journal Citation Reports y Scopus," *El Profesional de la Información*, vol. 19, no. 2, pp. 201–207, 2010.

[91] ScienceDirect, "What does Scopus cover? — SciVerse," 2012.

[92] T. Kohonen, *Self-Organizing Maps.* Springer, 3 ed., 2001.

[93] C. Blum and X. Li, *Swarm intelligence in optimization*, pp. 43–85. Springer, 2008.

[94] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micro machine and Human Science*, pp. 39–43, 1995.

[95] P. Novoa and C. Cruz, "Control de fallos en PSO Multi-Enjambre y su aplicación en ambientes dinámicos," in *Proceedings of the VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, (Málaga), 2009.

[96] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IV International Conference on Neural Networks*, pp. 1942–1948, 1995.

[97] M. Clerc, "Particle swarm optimization," 2006.

[98] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[99] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the 2000 Congress on Evolutionary Computation*, (La Jolla, CA), pp. 84–88, 2000.

[100] O. J. Alejo Machado, J. M. Fernández-Luna, and J. F. Huete-Guadix, "RankPSO: A New L2R algorithm Based on Particle Swarm Optimization," *Journal of Multiple-Valued Logic and Soft Computing (JMVLSC)*, vol. 23, no. 1-2, pp. 1–34, 2014.

[101] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, no. 6, pp. 417–441, 1933.

[102] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[103] P. E. Green, *Analyzing Multivariate Data*. 1978.

[104] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel pca and de-noising in feature spaces," in *Proceedings of the 1998 conference on Advances in Neural Information Processing Systems II*, vol. 11, pp. 536–542, 1999.

[105] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. 2 ed., 2005.

[106] P. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, no. 1, pp. 53–65, 1987.

[107] C.-m. Yan, B.-l. Guo, and X.-x. Wu, "Empirical Study of the Inertia Weight Particle Swarm Optimization with Constraint Factor," *Journal of Soft Computing And Software Engineering*, vol. 2, no. 2, pp. 1–8, 2012.

[108] L. Li and H.-T. Lin, "Ordinal regression by extended binary classification," in *Proceedings of the 20th Annual Conference on Neural Information Processing Systems*, pp. 865–872, 2006.

[109] K. Morik, P. Brockhausen, and T. Joachims, "Combining Statistical Learning with a Knowledge-Based Approach - A Case Study in Intensive Care Monitoring," in *ICML '99 Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 268–277, 1999.

[110] O. Chapelle and S. S. Keerthi, "Efficient algorithms for ranking with SVMs," *Information Retrieval Journal*, vol. 13, no. 3, pp. 201–215, 2010.

[111] O. Chapelle and M. Wu, "Gradient descent optimization of smoothed information retrieval metrics," *Information Retrieval Journal*, vol. 13, no. 3, pp. 216–235, 2010.

[112] D. W. Stockburger, *Multivariate Statistics: Concepts, Models, and Applications*. Missouri State University, 2nd ed., 2001.

[113] D. J. Sheskin, *Handbook of Parametric and Nonparameteric Statistical Procedures*. third edit ed., 2003.

[114] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.

[115] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[116] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[117] Y. Caballero, *Aplicación de la Teoría de los Conjuntos Aproximados en el Prepocesamiento de los Conjuntos de Entrenamiento para los Algoritmos de Aprendizaje Automatizado.* tesis doctoral, Universidad Central de las Villas, 2007.

[118] K. A. De Jong, *An analysis of the behaviour of a class of genetic adaptive systems.* Phd thesis, University of Michigan, 1975.

[119] O. J. Alejo Machado, J. M. Fernández-Luna, J. F. Huete-Guadix, and E. Concepción-Morales, "Fisherman Search Procedure," *Progress in Arti cial Intelligence. Special Issue: Metaheuristics Applied To Solve Real World Problems*, ISSN: 2192-6352, DOI 10.1007/s13748-014-0052-7, 2014.

[120] M. Molga and C. Smutnicki, "Test functions for optimization needs," in *http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf*, 2005.

[121] T. A. Feo and M. G. C. Resende, "Greedy Randomized Adaptive Search Procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[122] S. Yuhui and R. Eberhart, "A modified particle swarm optimizer," in *IEEE World Congress on Computational Intelligence*, (New York, USA), pp. 69–73, 1998.

[123] R. Storn and K. Price, "Differential Evolution A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[124] X. Geng, T.-Y. Liu, T. Qin, and H. Li, "Feature selection for ranking," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, USA), pp. 407–414, 2007.

[125] A. Y. Ng., "Feature selection, L1 vs. L2 regularization, and rotational invariance," in *Proceedings of the twenty-first International Conference on Machine learning*, (New York, USA), p. 78, 2004.

[126] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern information retrieval*. first ed., 1999.

[127] F. Pan, T. Converse, D. Ahn, F. Salvetti, and G. Donato, "Feature selection for ranking using boosted trees," in *Proceedings of the 18th ACM conference on Information and knowledge management*, (New York, USA), pp. 2025–2028, 2009.

[128] V. Dang and W. Croft, "Feature Selection for Document Ranking using Best First Search and Coordinate Ascent," in *Proceedings of SIGIR Workshop on Feature Generation and Selection for Information Retrieval*, 2010.

[129] P. Gupta and P. Rosso, "Expected Divergence based Feature Selection for Learning to Rank," in *Proceedings of the 24th International Conference on Computational Linguistics*, (Mumbai, India), pp. 431–440, 2012.

[130] H.-J. Lai, Y. Pan, Y. Tang, and R. Yu, "FSMRank: Feature Selection Algorithm for Learning to Rank," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 940–952, 2013.

[131] E. J. Mesa Delgado, *Supernova: un algoritmo novedoso de optimización global*. PhD thesis, Universidad Nacional de Colombia, 2010.

[132] N. Balasubramanian, *Query-Dependent Selection of Retrieval Alternatives*. Dissertations, University of Massachusetts - Amherst, 2011.

[133] Y. Suhara, J. Suzuki, and R. Kataoka, "Robust Online Learning to Rank via Selective Pairwise Approach Based on Evaluation Measures.," *Transactions of the Japanese Society for Artificial Intelligence*, vol. 28, no. 1, pp. 22–33, 2013.

[134] J. L. Elsas, V. R. Carvalho, and J. G. Carbonell, "Fast learning of document ranking functions with the committee perceptron," in *Proceedings of the international conference on web search and web data mining*, pp. 55–64, 2008.

[135] D. Sculley, "Large scale learning to rank," in *Proceedings of the NIPS'09 Workshop on Advances in Ranking*, 2009.

[136] D. A. Metzler, "Automatic feature selection in the markov random field model for information retrieval," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 253–262, 2007.

[137] L. Breiman and J. H. Friedman, "Estimating optimal transformations for multiple regression and correlation," *Journal of the American Statistical Association*, vol. 80, no. 391, pp. 580–598, 1985.

[138] O. J. Alejo Machado, J. M. Fernández-Luna, J. F. Huete-Guadix, and E. Moreno-Cerrud, "L2RLab: Integrated Experimenter Environment for Learning to Rank," in *Flexible Query Answering Systems - 10th International Conference, FQAS 2013*, pp. 543–554, Lecture Notes in Computer Science, Vol. 8132, 2013.

[139] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco: Morgan Kaufmann, second ed., 2005.

[140] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1536, 2001.

[141] D. Metzler and W. B. Croft, "Linear feature-based models for information retrieval," *Information Retrieval*, vol. 10, no. 3, pp. 257–274, 2007.

[142] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao, "Adapting Boosting for Information Retrieval Measures," *Information Retrieval*, vol. 13, no. 3, pp. 254–270, 2007.

[143] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[144] A. P. Bartés and X. T.-M. Llabrés, *Métodos estadísticos. control y mejora de la calidad*. 2005.

[145] M. D. Fernández, *Diseño de experimentos*. PhD thesis, Instituto Superior Politécnico José Antonio Echevarría, Cuba, 2004.

[146] P. Novoa, D. A. Pelta, and C. Cruz, "DynOptLab: una herramienta para la experimentación en ambientes dinámicos," in *Tendencias de Soft Computing*, Editorial Feijóo, UCLV, 2009.

[147] D. C. Montgomery, *Design and analysis of experiments*. 5th editio ed., 2000.

[148] T. Bartz-Beielstein, *Experimental Research in Evolutionary Computation: The New Experimentalism*. New York, USA: Springer Publishing Company, 2006.

[149] M. E. Fayad, D. C. Schmidt, and R. E. Johnson, *Building application frameworks: Object oriented foundations of framework design*. 1 ed., 1999.

[150] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[151] P. B. Nemenyi, *Distribution-free multiple comparisons*. 1963.

[152] V.-d. Cung, S. L. Martins, C. C. Ribeiro, and C. Roucairol, "Strategies for the parallel implementation of metaheuristics," in *Essays and Surveys in Metaheuristics*, pp. 263–308, Kluwer Academic, 2002.