Universidad de Cienfuegos "Carlos Rafael Rodríguez"
Facultad de Ingeniería
Carrera de Ingeniería Informática

"Estudio experimental del aprendizaje de la ordenación utilizando técnicas de reducción de dimensionalidad de datos."

Trabajo de diploma para optar por el título de Ingeniería en Informática

Autor:
Omar Ernesto George Farray

Tutor:
Msc. Oscar José Alejo Machado

Cienfuegos, Cuba
Curso 2013-2014

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al Departamento de Informática de la Facultad de Ingeniería en la Universidad de Cienfuegos “Carlos Rafael Rodríguez”, para que hagan el uso que estimen pertinente del trabajo de diploma.

Para que así conste firmo la presente a los 4 días del mes de junio del 2012.

Omar Ernesto George Farray

Nombre completo del autor

Msc. Oscar José Alejo Machado

Nombre completo del tutor

Los abajo firmantes certificamos que el presente trabajo ha sido revisado según acuerdo de la dirección de nuestro centro y el mismo cumple los requisitos que debe tener un trabajo de esta envergadura referente a la temática señalada.

Firma Tutor

Firma ICT

Firma Vicedecano

Agradecimientos

A mi familia toda.

A mis padres y hermana, cuyo constante aliento y exigencia me trajeron hasta este resultado.

A mi novia, secretaria, consultante y compañera en las largas jornadas de intenso trabajo.

A mi tutor, distante pero cercano en su esfuerzo por ayudarme a materializar este empeño.

Dedicatoria

A todas las personas, cuyos consejos, conocimientos o apoyo, contribuyeron a la realización de este trabajo. Son tantas que no alcanzaría a mencionarlas en esta sola cuartilla.

Resumen

Learning to Rank (L2R) es un área de investigación multidisciplinaria que abarca las ramas de la Recuperación de Información (RI) y el Machine Learning (ML). Específicamente en RI, el problema clave del L2R 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 o rasgos.

Desarrollar modelos de ranking que tengan un impacto positivo a la hora de mostrar los resultados de una determinada búsqueda, constituye una tarea compleja que ha atraído el interés de la comunidad científica y la incorporación constante de nuevas técnicas de ML, y más recientemente de estrategias basadas en Soft Computing.

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.

A partir de estas premisas, se introduce una variante adaptada y mejorada de FSP para tratar el problema del L2R. Este método, denominado RankFSP, es capaz de construir una función de ranking optimizando cualquier medida de RI, e implementa además una estrategia para evadir los mínimos locales y los puntos de búsqueda estancados. También se propone un nuevo método de selección de rasgos para reducir la dimensionalidad de las colecciones de datos estudiadas. Todas estas propuestas fueron validadas experimentalmente, demostrando con su aplicación mejoras sustanciales en la tarea del ranking.

Índice de contenido

Índice de tablas

Índice de figuras

Introducción

Los primeros intentos del hombre para almacenar datos -entiéndase esto como almacenamiento físico de la información- comenzaron con los primeros habitantes del planeta, quienes plasmaban en diferentes soportes como piedra, madera y arcilla una simbología muy rudimentaria de fenómenos naturales, prácticas cotidianas, como la caza de animales, el apareamiento y otros sucesos.

Esto se debía, fundamentalmente, a la limitación del discurso oral para preservar el registro de los acontecimientos históricos y a la necesidad de llevar esa información a grandes distancias, para transmitirla más allá del tiempo y del espacio.

La modalidad escrita para la transmisión de información, surgió luego como respuesta de las clases dominantes a la necesidad de consolidar su estatus. En toda la Antigüedad y la Edad Media, la información oral y fundamentalmente la escrita, constituyeron las herramientas principales de poder de esas clases para afianzar sus privilegios y someter a sus subordinados.

La escritura constituyó un paso significativo en la búsqueda de recursos para plasmar y almacenar la información fuera de la mente humana, y creó los cimientos tecnológicos para hacer trascender la memoria colectiva.

En el siglo V surgen las primeras bibliotecas en Pérgamo y Alejandría. Comienzan a circular libros en Grecia a partir de librerías talleres cuyos dueños confeccionaban, vendían y exportaban los manuscritos incluso a otros países. Son las primeras instituciones ideadas por el hombre con el fin de preservar la información escrita, registrada en determinados soportes.

Con el nacimiento de las universidades, siglos después, el pergamino ya no constituyó más el principal soporte para la escritura. La información adquirió una mayor portabilidad con la introducción del papel. El comercio del libro, ahora menos lujoso, volvió a tomar vida con precios más reducidos, aunque aún costosos, por lo que su adquisición era todavía privilegio de unos pocos.

Los incipientes pasos tecnológicos operados en la manufactura durante ese período, facilitaron el surgimiento de otra modalidad de intercambio de la información basada en una nueva técnica: la imprenta.

Con ese invento se inició la difusión masiva de información, que aún en esta etapa se controlaba y pasaba por el filtro de las clases dominantes. Este episodio constituyó, sin dudas, el punto de partida para compartir la información de manera más participativa y masiva.

Relacionados con la imprenta, en siglos posteriores se sucedieron varios adelantos socio-tecnológicos que agilizaron en gran medida el progreso de la información: se generalizó el procedimiento de fabricación del papel y la impresión de la prensa a partir de máquinas de vapor. Las publicaciones periódicas se convirtieron en los instrumentos idóneos para la transferencia del conocimiento y surgieron las revistas científicas como expresión de un mayor grado de especialización. No obstante la información seguía siendo muy dependiente de los medios de transporte de la época.

Tal situación varió sustancialmente con el advenimiento del siglo XX y el auge de las industrias de servicios, que permitieron el desarrollo de una nueva vertiente: los servicios de información.

Terminada la Segunda Guerra Mundial aumentaron con rapidez las publicaciones e informaciones en general y con ellas el conocimiento y la especialización. Se desarrollan a la par la información y la cibernética. Ya en el año 1946 apareció la primera computadora y un cuarto de siglo después surgieron los microordenadores. Los cambios que acontecieron a partir de entonces, generaron la llamada "Era de la Información", en la cual la industria de la información era sinónimo de riqueza y poder.

Consecuente con una tendencia mantenida a través de la historia, la utilidad de los soportes de información también está hoy en función de lograr una mayor capacidad de almacenamiento y perdurabilidad. Estos han evolucionado a tal punto que la información, antes plasmada en piedra, ahora se conserva en medios electrónicos. Los primeros soportes de este tipo fueron las tarjetas perforadas, sustituidas más tarde por los discos flexibles y compactos, para almacenar una mayor cantidad de información en un espacio menor.

Pero esa dimensión ya no importa en el mundo vertiginoso de hoy. Las redes permiten transmitir información sin desplazamientos físicos. La memoria del mundo

ya no está sólo en las grandes bibliotecas porque por medio de una base de datos las personas pueden acceder a grandes volúmenes de información.

Hoy los flujos de información no reconocen fronteras nacionales, impulsados por el libre comercio, en una sociedad globalizada, cuya manifestación más evidente es la supercarretera de la información: INTERNET.

La red de redes, como también se le conoce, dispone de un amplio número de recursos que permiten, mediante las páginas web y otras aplicaciones, acceder de manera fácil a informaciones en texto, imagen, audio y vídeo.

Pero cada día este volumen crece notablemente, por lo que sería complicado dar cifras exactas sobre la cantidad de datos que se mueven a diario en el ciberespacio.

Con la primera explosión de Internet, comenzó a consolidarse un gran almacén de información universal: un Big Bang que ha provocado que, al igual que el universo, la red de redes viva en un permanente estado de expansión. Prueba de ello es que en tan solo sesenta segundos se generan aproximadamente más de 204 millones de mensajes; 2 millones de consultas de dudas en Google; la subida de 48 horas de vídeo a la plataforma YouTube; la muestra de casi 685.000 fotos y contenidos en Facebook o la publicación de 100.000 tuits en Twitter[1]. En estas últimas décadas se han generado más datos que los producidos por la humanidad en toda su historia anterior. Y esta explosión no ha hecho más que comenzar.

Con la proliferación de Internet, como medio principal de comunicación electrónica de datos, las personas han comenzado a lidiar con grandes y diversos volúmenes de información. Usarlos, manipularlos y ordenarlos resulta una tarea en extremo compleja y costosa, que ha llamado la atención a la comunidad científica.

Desde el año 1983, Gerard Salton proporcionó la definición más completa de este campo de investigación denominado *Recuperación de Información* (RI)[2] al que describió como un campo relacionado con "*la representación, almacenamiento, organización y acceso a los ítem de información*".[1]

[1] Información publicada el 13/02/2013 en el sitio web *LeaNoticias.com* por la compañía americana especializada en software DOMO, a partir de una infografía referenciada como "Data Never Sleeps".

[2] En otras publicaciones aparece indicado como *IR* por su terminología en inglés: *Information Retrieval*. En esta investigación utilizaremos indistintamente ambas referencias.

Para satisfacer las necesidades de cada usuario se crearon disímiles sistemas de recuperación de información tanto para la web visible como Google, Yahoo, Bing, entre otros, como para la web invisible o internet profunda[3].

Como esta tarea se ha venido complejizando con el paso del tiempo, se ha hecho necesaria la introducción de técnicas de inteligencia artificial para lograr una mayor precisión y eficiencia del proceso de ordenación. Este campo ha cobrado fuerza en pocos años debido a los buenos resultados obtenidos de la alianza entre la Recuperación de Información y el Aprendizaje Automático de donde surge una nueva área de investigación: *Learning to Rank* (L2R)[4].

Los métodos de este Aprendizaje de la Ordenación están basados en técnicas de aprendizaje automático supervisado, con la meta de aplicar un modelo de recuperación a los datos de entrenamiento para predecir, con la mayor exactitud, los juicios de relevancia a partir de una función de pérdida o error. Estas colecciones de entrenamiento o *training data* -como se les conoce en inglés- se componen de un conjunto de consultas y documentos asociados a ellas, con un orden por lo general dictaminado por una puntuación numérica o por un juicio binario que determina entre "relevante" o "no relevante" a cada elemento de la colección. Estos conjuntos de datos son usados por algoritmos de aprendizaje para producir un modelo de ordenación. El modelo obtenido se probará con un nuevo juego de consultas y será reemplazado de acuerdo con la predicción de relevancia, por una nueva clasificación y re-ordenación de esos documentos.

Hoy cada persona en el mundo es potencialmente portadora de información que puede difundir en redes sociales, blogs o páginas en la web. El único problema es que mucha de esa información es irrelevante o está repetida.

Filtrarla y permitirnos ver con más facilidad sólo lo más valioso para nuestros intereses y prioridades, es una de las grandes dificultades que presentan los modelos de ranking en la Recuperación de Información. El conflicto radica en hallar

[3] La causa principal de la existencia de la *Internet Profunda* es la imposibilidad de los motores de búsqueda para encontrar o indexar toda la información existente en Internet.

[4] Este término tiene variantes en su traducción al español. Algunos autores lo traducen como Aprender a Rankear, mientras otros se refieren a él como Aprendizaje de la Ordenación (este último es el que usaremos en nuestra investigación de tesis).

correspondencia entre los términos empleados por los usuarios en sus consultas y los documentos de los autores en la web. Entonces, ¿cómo conciliar búsquedas con resultados apropiados y preguntas con respuestas correctas?

El Aprendizaje de la Ordenación no sólo puede limitarse a devolver documentos relevantes para la consulta realizada, sino que debe mostrarlos en un orden de relevancia. Por lo tanto, con este estudio teórico-práctico se pretende profundizar en el campo de la IR en la web, enfatizando en la recuperación de documentos y aportando nuevos métodos en los modelos de L2R aplicados a grandes colecciones de entrenamiento, para lograr la mayor cantidad posible de documentos relevantes, minimizando o manteniendo el tiempo de respuesta.

Por lo que la tesis define como **problema a resolver:**

¿Cómo disminuir el tiempo de respuesta de los modelos de L2R (Learning to Rank) sobre grandes conjuntos de entrenamiento manteniendo o mejorando la precisión referenciada en la literatura?

Se identifica como **objeto de estudio** de este trabajo de investigación: los métodos de L2R que optimizan directamente medidas de desempeño.

El **campo de acción** sobre el que se trabajará son las técnicas de reducción de dimensionalidad en los métodos de L2R.

Encontrar una estrategia capaz de mejorar las soluciones y el tiempo de respuesta de los modelos de L2R, es una tarea compleja que sugiere el uso de técnicas computacionales basadas en SoftComputing (Algoritmos Evolutivos, Programación, Genética, Lógica difusa, entre otros).

A partir del problema planteado se precisa como **Idea a defender:** la utilización de una nueva estrategia para la reducción de dimensionalidad en grandes colecciones estándar de datos para el L2R, que permitirá disminuir el tiempo de respuesta.

De este modo se definen como **objetivo general**: Proponer una estrategia que mejore el tiempo de respuesta de L2R frente a grandes colecciones de datos.

De este objetivo general se han deducido los siguientes **objetivos específicos**:

- Diseñar una estrategia para la reducción de dimensionalidad de los datos en las grandes colecciones que se utilizan en L2R.

- Comparar los resultados obtenidos del método RankFSP con otros métodos referenciados en la literatura.

Las **tareas científicas** a realizar para cumplir los objetivos propuestos son:

- Estudio de las principales categorías, modelos y artículos científicos publicados en el área de investigación del Aprendizaje de la Ordenación para la R.I.
- Estudio de los procedimientos de reducción de dimensionalidad aplicados al L2R.
- Implementación de un procedimiento de reducción de rasgos aplicado a la tarea del L2R.
- Determinación del rendimiento que se puede alcanzar con el modelo RankFSP y los procedimientos propuestos mediante su evaluación con grandes colecciones documentales estándar.
- Comparación de los desempeños obtenidos con respecto a otros modelos referenciados en L2R, así como con los que optimizan directamente medidas de desempeño utilizadas en R.I.

La memoria gráfica de este estudio está estructurada de la siguiente forma: una introducción con elementos de aproximación al tema investigado donde se ubica al lector en el problema a resolver, su contexto y otras cuestiones introductorias. Tres capítulos: el primero donde se abordan los sustentos teóricos del estudio; el segundo con la fundamentación metodológica del mismo y un tercero donde aparece el producto de la investigación, es decir, los resultados. Elaboramos conclusiones y recomendaciones en respuesta a los objetivos planteados y sin paginar aparecen referencias bibliográficas y anexos que develan el rigor científico del estudio.

Capítulo 1: Estado del Arte del Aprendizaje de la Ordenación

1.1 Introducción

En este capítulo se aborda de forma panorámica y conceptual, las principales características y los objetivos que persigue el aprendizaje de la ordenación como problema clave en la recuperación de documentos, partiendo de una serie de conceptos básicos que diversos autores han venido definiendo sobre la Recuperación de Información, así como, la descripción de las principales categorías y modelos en general. Finalmente se presentan las conclusiones.

1.2 Recuperación de Información

La Recuperación de Información (RI) comienza a desarrollarse desde finales de la década de los años cincuenta, cuando el programador e informatólogo estadounidense Calvin N. Mooers introduce por vez primera el término *information retrieval*, refiriéndose a este como a una búsqueda de información en un almacén de documentos, efectuada a partir de la especificación de un tema.[2]

Su creciente importancia en la actualidad se debe a que el volumen de información existente se incrementa permanentemente, desde simples archivos de texto o un periódico electrónico, hasta librerías digitales y espacios mucho más grandes y complejos.

Por lo que resulta primordial tratar con toda esa información de carácter electrónico mediante su representación, almacenamiento, organización y acceso, que según Gerard Salton definen el concepto de Recuperación *de Información*.

No obstante hay elementos que conspiran contra la claridad de ese término. Su uso inapropiado ha hecho que el mismo no se encuentre bien empleado en muchas ocasiones. Respecto a esto, el profesor Charles T. Meadow expone que el concepto

de selectividad está implícito en la RI, ya que "*information recovery*[5] no es lo mismo que *information retrieval,* al menos que haya habido un proceso de búsqueda y selección."[3]

Por su parte, David A. Grossman acota este concepto cuando plantea que: "*IR está dedicada a encontrar documentos relevantes, no simples coincidencias en los modelos.*"[4]

Un documento recuperado es relevante cuando satisface una determinada consulta. Y un documento puede considerarse relevante si en su contenido aparece alguna información trascendente o importante con respecto a la pregunta realizada por el usuario. Por tanto, la relevancia de un documento recuperado es la relación que existe entre los contenidos de este y las peticiones explícitas de una consulta.

J. Perez-Carballo y Strzalkowski hacen la descripción más clara y espontánea de un proceso básico de Recuperación de Información, cuando señalan que es "*una típica tarea de traer documentos relevantes desde un gran archivo en respuesta a una pregunta formulada por el usuario y ordenar estos documentos de acuerdo a su relevancia*."[5]

Disponer o no de información rápida y precisa puede resultar en el éxito o fracaso de una operación. De ahí la importancia de los Sistemas de Recuperación de Información (SRI) para manejar estas situaciones de manera eficaz y eficiente.

Se impone entonces elegir en la presente investigación un modelo para el diseño de un SRI a partir de las definiciones anteriormente propuestas, aunque antes resulta necesario precisar qué es un "Sistema de Recuperación de Información".

A propósito, Salton piensa que "cualquier SRI puede ser descrito como un conjunto de ítems de información, un conjunto de peticiones y algún mecanismo que determine qué ítem satisface las necesidades de información expresadas por el usuario en la petición"[1], aunque el mismo autor reconoce que en la práctica tal procedimiento resulta muy simple y requiere ser ampliado.

Si tenemos en cuenta que el diseño de un SRI se realiza bajo un modelo que define la estrategia para evaluar la relevancia de un documento respecto a una consulta y

[5] Sin embargo cuando este término se traduce al español adquiere la misma connotación que *information retrieval,* aunque conceptual y tecnológicamente no signifique lo mismo.

de métodos para establecer un orden de los documentos recuperados, es necesario estudiar primero los modelos convencionales de ordenación para recuperar información, así como los métodos desarrollados para este fin.

1.3 Modelos convencionales de ranking para R.I.

En las ciencias de la computación existe un área, la Recuperación de Información, que ha sido tratada y diseñada a partir de modelos convencionales de ranking. Estos modelos tradicionales definen una función de recuperación u ordenación para asociar un grado de relevancia con un documento y una consulta dados.

Los principales modelos convencionales de ranking para R.I. se encuentran enmarcados específicamente en dos vertientes principales: los modelos dependientes de la consulta y los modelos independientes de la consulta.

1. **Modelos dependientes de la consulta.**
   - Modelo Booleano (*Boolean Model*) [6], Modelo Booleano Extendido [7].
   - Modelo del Espacio Vectorial (*Vector Space Model*) [8].
   - Modelos Probabilísticos (*Probabilistic Models*) [9].
   - Modelo BM25 [10].
   - Modelo del lenguaje estadístico (*Statistical language model*) [11] [12].
   - *Latent Semantic Indexing* (LSI) [13].
   - Modelos del lenguaje para la R.I. (*Languaje Models for Information Retrieval*, LMIR) [14] [15] [16].

Los primeros modelos de consulta-dependiente recuperaban documentos basados en las ocurrencias de los términos de consulta, como por ejemplo los modelos booleanos. Básicamente estos modelos pueden predecir si un documento es relevante a la consulta o no, pero no pueden predecir el grado de relevancia. Para esto entonces fue propuesto el modelo vectorial, hoy en día el más popular para la recuperación de información.

2. **Modelos independientes de la consulta**
   - PageRank [17].

- TrustRank [18].
- BrowseRank [19].

Los modelos de consulta-independiente clasifican sus documentos haciendo uso de la estructura de enlace de la web. El PageRank[6], por ejemplo, es el modelo probabilístico independiente a la consulta más empleado en la mayoría de los motores de búsquedas para la recuperación de información.

En esta etapa de grandes retos científicos, los referidos modelos no solo sentaron una importante pauta, sino que trascendieron, directa o indirectamente, en las concepciones iniciales de la mayoría de las investigaciones que actualmente se realizan en el campo de L2R y la R.I.

A pesar de estas consideraciones, tales modelos convencionales se enfrentan a las siguientes limitaciones:

- Ajustar los parámetros de forma manual es usualmente difícil, sobre todo cuando son muchos y las medidas de evaluación son *non-smooth*[7].
- 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.

Solucionar tales problemas requiere del uso de técnicas y estrategias de Aprendizaje de Máquinas (ML)[8] como herramientas efectivas que permitan:

- Ajustar los parámetros de forma automática.
- Combinar múltiples evidencias.
- Evitar el sobreajuste.

[6] Este modelo decide la valoración de las páginas mediante enlaces que significan votos: si una página enlaza con otra, considera que está dando un voto a esa página que vincula. Según el número de votos (o enlaces) recibidos por una página, su posición variará. A mayor número de votos, mejor posición entre los resultados.
[7] Término inglés que se refiere en dicho contexto a una función discontinua y no diferenciable.
[8] *(ML)* por su traducción al inglés: Machine Learning.

1.4 Descripción del problema

Existen muchas indicaciones que pueden influir en la relevancia de un documento en una página web. Ejemplo de ello son los textos de anclaje, los clics de datos guardados en los registros de búsquedas y la puntuación de los modelos tradicionales de ranking tales como BM25 o PageRank. Estos últimos requieren de una elección de parámetros óptimos, que inciden en la problemática del sobreajuste. A esto, se le suma también el inconveniente de cómo combinar diferentes modelos de ordenación.

La incorporación de dicha información en las técnicas de aprendizaje automático, ha provocado la vinculación del Aprendizaje de Máquinas (ML) a los campos de la RI. Las tecnologías de ML tienen como objetivo desarrollar técnicas para que las máquinas se instruyan por si solas. Su función principal es construir automáticamente un modelo de clasificación de texto, apoyado en el aprendizaje de las características de una colección de documentos ya clasificados con anterioridad. La ventaja que presenta esta rama de la Inteligencia Artificial *"es que entrega resultados en forma automática y exacta, aliviando los juicios de relevancia que deben realizar los usuarios expertos a cada documento."*[20]

Para contribuir a ese propósito, evitar los sobreajustes y adaptar mejor distintos modelos de ranking, surge una nueva área de investigación denominada "Aprendizaje de la Ordenación" o "Learning to Rank".

El proceso básico del L2R se inicia a partir de una colección de entrenamiento, que no es más que un conjunto de consultas Q_N, asociadas cada una de ellas a sus documentos, representadas a su vez por un vector de atributos *d* y un juicio de relevancia *r*. A partir de esta colección se entrena un modelo de ranking que construye una función de ordenamiento: *f*. Luego, la función aprendida es aplicada sobre una nueva colección de prueba, generando una predicción de ranking de acuerdo con la consulta y sus documentos correspondientes.

En Figura 1.1 se ilustra el esquema típico del proceso del Aprendizaje de la Ordenación.

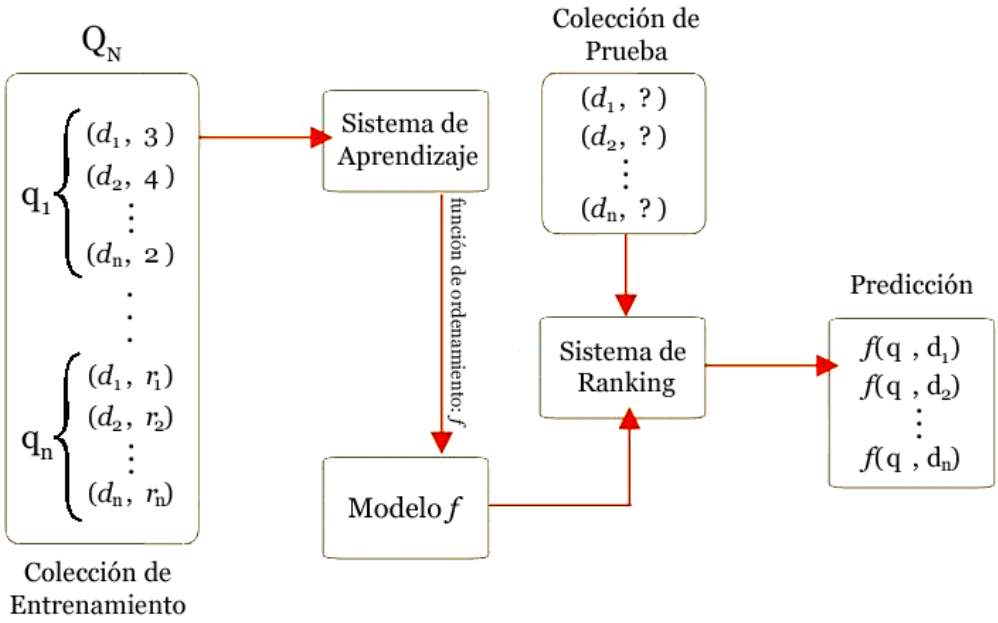


Figura 1.1: Proceso de Learning to Rank.

1.5 Principales categorías y modelos

Para desarrollar un modelo de ranking con las técnicas de Aprendizaje Automático para la Recuperación de Información se parte de los siguientes componentes claves[21]:

- **Espacio de Entrada** o *Input Space*: contiene los objetos de estudio representados generalmente por vectores de atributos, extraídos de las diferentes aplicaciones.

- **Espacio de Salida** u *Output Space*: su objetivo es aprender en base a los objetos de entrada.

- **Espacio de Hipótesis** o *Hypothesis Space*: define las funciones que operan y transforman los vectores de los objetos de entrada, en predicciones de acuerdo a los objetos de espacio de salida.

- **Función de Pérdida** o *Loss Function*: mide e indica el grado de predicción generado por los resultados de las funciones del espacio de hipótesis respecto a salidas.

En base a estos cuatro elementos del Aprendizaje Automático, anteriormente definidos, se consideran tres categorías de aproximaciones al problema[20]:

- **Aproximación por puntos**
  - *Espacio de entrada*: contiene el vector con las características de cada documento en forma individual.
  - *Espacio de salida*: representa el grado de relevancia de cada documento.
  - *Espacio de hipótesis*: contiene funciones que toman las características del vector de un documento como entrada y predicen el grado de relevancia del documento respecto a la consulta.
  - *Función de pérdida*: examina la precisión del valor real para cada documento clasificado.

- **Aproximación por pares**
  - *Espacio de entrada:* contiene un par de documentos, ambos representados por sus vectores de atributos.
  - *Espacio de salida:* contiene el grado de relevancia entre cada par de documentos.
  - *Espacio de hipótesis:* contiene funciones bi-variables capaces de ordenar correctamente pares de documentos respecto a su relevancia a partir de una consulta.
  - *Función de pérdida:* mide la inconsistencia, es decir, cuantifica el número de pares de documentos mal ordenados.

- **Aproximación por listas**

- *Espacio de entrada:* contiene un grupo entero de documentos relacionados con una consulta.
- *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, para predecir sus grados de relevancia.
- *Función de pérdida:* mide la diferencia entre la lista ordenada dada por la hipótesis y la lista de predicción original de cada documento.

1.5.1 Aproximación por puntos (Pointwise Approach)

Cuando usamos técnicas de Aprendizaje Automático para resolver problemas de ordenación, la sencillez es lo que significa a esta categoría por encima de las otras. Los modelos de esta aproximación simplemente aplican algoritmos previamente desarrollados, para traducir los juicios de relevancia a valores numéricos.

Sin embargo, la debilidad de los métodos expuestos reside en su propia sencillez. Esto es porque, conceptualmente, las tareas de regresión, clasificación y ranking son muy distintas y se requiere de métodos más específicos que consideren relaciones de orden.[22]

A continuación se explican los algoritmos más representativos en tres subcategorías[20]:

1. *Algoritmos basados en Regresión*: en esta subcategoría se considera el grado de relevancia como números reales. A continuación se presentan algunos ejemplos:
    - Función de Regresión Polinomial.
    - Subconjunto de Ranking con Regresión.

2. *Algoritmos basados en Clasificación*: estos algoritmos no consideran la etiqueta *ground truth*[9] como un valor cuantitativo. A continuación se presentan algunos ejemplos:
    - Modelo Discriminativo para IR.
    - Clasificación Multi-Clase para Ranking (McRank).

3. *Algoritmos basados en Regresiones Ordinales*: estos toman la relación ordinal entre las etiquetas *ground truth* durante el proceso de aprendizaje del modelo de ordenación. A continuación se presentan algunos ejemplos:
    - Ranking basado en Perceptron (PRanking).[23]
    - Ranking con Principios de Margen Grande.

1.5.2 Aproximación por pares (Pairwise Approach)

Esta categoría de aproximación consiste en seleccionar pares de documentos, cuya relevancia relativa con respecto a una consulta ya sea conocida con anterioridad. A partir de estos pares, se formulan varios algoritmos que operan con el orden relativo de estos documentos y sus espacios de entrada.

Algunos de estos modelos tienen la ventaja de adaptar métodos clásicos de Aprendizaje Automático a partir de la consideración de diferencias en el espacio de entrada o de salida; siendo esta adaptación consistente y coherente con el planteamiento a pares del problema.

A continuación se nombran los algoritmos más característicos:

1. Algoritmos Ordenados con Función de Preferencia.
2. RankNet.[24]
3. FRank.[25]

[9] Se traduce al español como: la realidad tal y como es. Pero en el caso de etiquetas *ground truth*, hace referencia al valor real del documento clasificado.

4. RankBoost.[26]
5. Ranking SVM (Máquina de Soporte Vectorial).[27]

1.5.3 Aproximación por listas (Listwise Approach)

La aproximación por listas se puede dividir en dos subcategorías. Una primera, donde el espacio de salida contiene los grados de relevancia de todos los documentos relacionados con una consulta, y la función de pérdida es definida en la aproximación o límites de las medidas de evaluación en IR. En la otra segunda subcategoría, 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*.
A continuación se presentan las dos subcategorías con sus algoritmos más representativos:

- *Minimización de las Pérdidas en Listwise de Ranking*: en esta subcategoría, se minimiza la función de pérdida definida sobre las inconsistencias entre la salida del modelo de ranking y la permutación de las etiquetas *ground truth* de todos los documentos. A continuación se nombran algunos ejemplos:

  1. ListNet.[28]
  2. ListMLE.[29]

- *Optimización Directa de las Medidas de Evaluación en IR*: como lo surgiere el nombre se trata de optimizar una continua y diferenciable aproximación de una medida de evaluación en IR, y luego usar tecnologías de optimización que permiten perfeccionar objetivos más complejos.
  A continuación se nombran algunos ejemplos:

1. SoftRank.[30]
2. SVMmap.[31]
3. AdaRank.[32]
4. Algoritmos basados en Programación Genética.

Como esta subcategoría ha despertado el interés de muchos investigadores, algunos de ellos han desarrollado importantes trabajos que han agrupado a su vez en tres nuevas categorías dentro del aprendizaje de la ordenación:

1. Minimizar una función de pérdida que acote superiormente a una función de pérdida básica, definida sobre las medidas de R.I.
2. Aproximar las medidas de desempeño de R.I. mediante funciones fáciles de manipular.
3. Usar tecnologías especialmente diseñadas para optimizar medidas de desempeño de R.I. no suaves (*non-smooth*).

1.6 Métodos desarrollados

En la Figura 1.2, se muestra los principales métodos desarrollados desde el año 2002 hasta el 2013 en cada una de las categorías, donde se aprecia que en estos últimos años ha habido una tendencia a desarrollar métodos de L2R con características sustentadas en el enfoque por lista, un propósito que por cierto comparte esta investigación.

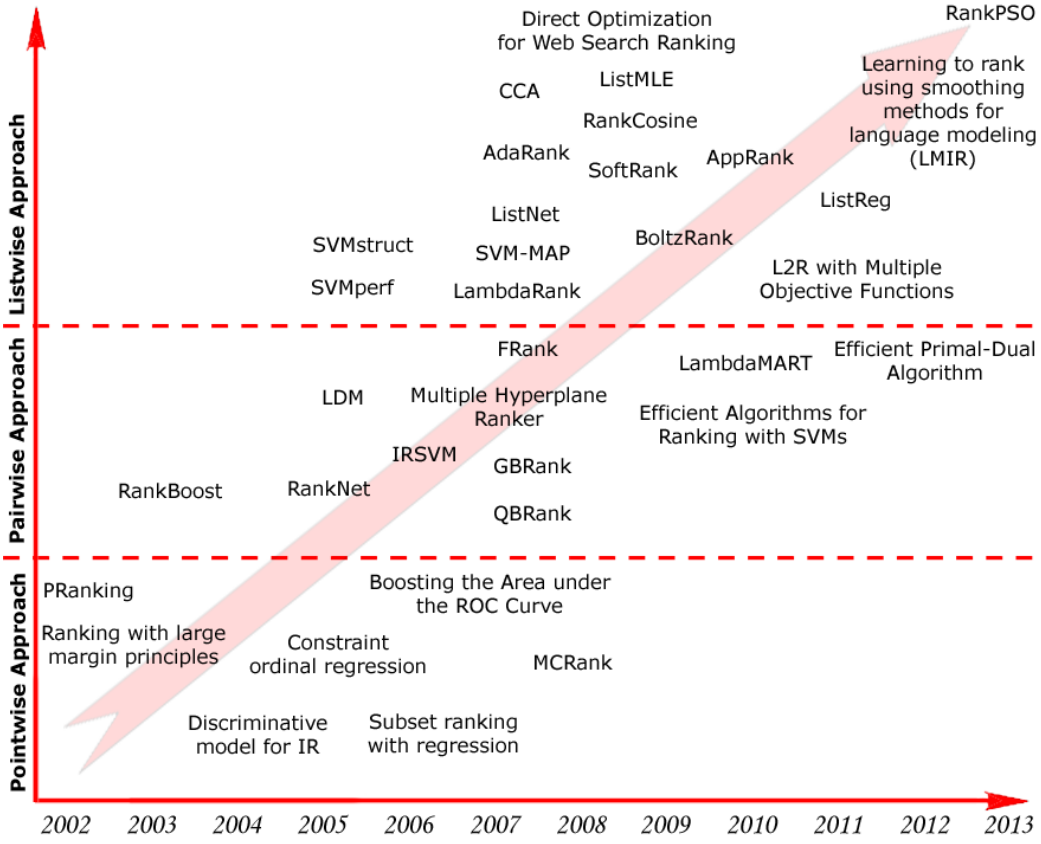


Figura 1.2: Línea de tiempo y tendencia categórica del Aprendizaje de la Ordenación.

1.7 Colecciones de datos utilizadas

Las colecciones de datos de LETOR[10], 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 que contemplan las colecciones OHSUMED, Gov, Gov2, MQ2007 y MQ2008. Más recientemente, en junio de 2010, *Microsoft Research*[11] liberó dos colecciones de referencia de gran

[10] Dirección web http://research.microsoft.com/en-us/um/beijing/projects/letor/
[11] Disponible en la URL: http://research.microsoft.com/en-us/projects/mslr/

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*[12], tienen 5 valores de 0 (irrelevante) a 4 (perfectamente relevante). Cada par consulta-documento 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*[13], o en el Anexo 1, donde también aparecen referenciados. La presente investigación está encaminada a abordar la tarea del L2R sobre estas últimas colecciones.

1.8 Medidas de evaluación

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:
    - Precision at n (P@n)[33].
    - Mean Average Precision (MAP)[33].
- Con múltiples niveles de relevancia:
    - Normalized Discounted Cumulative Gain (NDCG)[33].

Típicamente se emplean estas tres medidas de RI para evaluar la precisión en el ordenamiento. Estas son definidas de la siguiente manera:

1.8.1 Precision at n (P@n)

La precisión en *n* mide la relevancia de los documentos recuperados con

[12] Dirección: https://www.bing.com/
[13] Disponible en http://research.microsoft.com/en-us/projects/mslr/feature.aspx

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 devueltos a partir de una consulta son (*relevante*, *irrelevante*, *irrelevante*, *relevante*, *relevante*, *relevante*, *irrelevante*, *irrelevante*, *relevante*, *relevante*), entonces los valores de P@1 a P@10 serán (1, 1/2, 1/3, 2/4, 3/5, 4/6, 4/7, 4/8, 5/9, 6/10), respectivamente.[33]

1.8.2 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 559.[33]

1.8.3 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. Esta medida basada en la métrica DCG[10], pero normalizada por el ranking ideal de los resultados relevantes, mientras evalúa una lista de ordenamiento, sigue dos reglas fundamentales:

1. Los documentos de gran relevancia 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.

El valor de esta métrica para una lista de documentos dada, se calcula:

$$NDCG@y = \frac{1}{N} \sum_{i=1}^{y} \frac{2^{rel(d_i)} - 1}{\log(i+1)}$$

...donde *y* es la posición hasta la que se evalúa esta lista, *N* es el valor DCG ideal para los resultados relevantes (suponiendo que todos los resultados relevantes se presentan consecutivamente desde la primera posición del ranking), *i* es la posición del ranking de la lista de documentos que se está

evaluando, d_i es el documento en la posición i, y rel(d_i) es el valor de relevancia de d_i, siendo igual a 0 si el resultado es no relevante e igual a 1 si lo es.[11]

Existen otras medidas de evaluación dentro del ámbito de RI como ERR (Expected Reciprocal Rank)[12], RBP (Rank Biased Precision)[13], WTA (Winners Take All)[14], MRR (Mean Reciprocal Rank)[14], entre otras, que también han sido usadas pero de forma muy puntual, tanto en trabajos teóricos como prácticos, en el campo del L2R.

1.9 Fundamentación de la metodología utilizada.

Para el desarrollo de la investigación se emplea una metodología mixta ya que se utilizan elementos de la metodología cuantitativa y cualitativa. El enfoque cualitativo permite la interpretación del fenómeno y el enfoque cuantitativo, su explicación. Esta es una tendencia que se está utilizando cada vez con mayor frecuencia en las investigaciones.

La investigación cualitativa propone la estancia prolongada en el campo y la observación persistente de los focos principales de la investigación. Se recogen las informaciones y luego se analizan desde distintos ángulos a fin de contrastarlos, realizando el cruzamiento e interpretando y hallando las coincidencias de los resultados alcanzados.

Se consideran de utilidad los métodos cuantitativos, ya que los números constituyen instrumentos de modelación de la realidad, de aproximación al conocimiento y contribuyen a ordenar la información en el proceso de producción del conocimiento. La metodología mixta posibilita aprovechar las tendencias, la direccionalidad que pueden mostrar algunos métodos cuantitativos sobre la base del análisis cualitativo, para cruzar y verificar la información obtenida aplicando diferentes métodos. Por eso se considera que la aproximación a la verdad está vinculada con la pluralidad metodológica, ya que el método debe poseer la capacidad de reflejar el movimiento, la dinámica, la esencia del objeto y esto no se logra sin la participación

compensatoria de los diversos métodos existentes.

Entre los métodos cualitativos y cuantitativos empleados en este trabajo se encuentran la modelación algorítmica, la experimentación y el análisis estadístico.

Los métodos científicos, según su etimología, son el camino hacia el conocimiento. En la presente investigación se utilizan varios de estos métodos:

- Nivel teórico.
  - ✓ Inducción – deducción: La inducción expresa el movimiento de lo particular a lo general, o sea se llega a generalizaciones partiendo del análisis de casos particulares, mientras la deducción expresa el movimiento de lo general a lo particular. La combinación de ambos permite estructurar el conocimiento científico obtenido en la revisión bibliográfica.
  - ✓ Histórico – lógico: El método lógico debe basarse en los datos que proporciona el método histórico, de manera que no se convierta en un simple razonamiento especulativo. De igual forma, lo histórico no debe limitarse a la simple descripción de los hechos, sino explicarlos a partir de la lógica de su desarrollo. Este método da la posibilidad de conocer el problema de la ordenación en la Recuperación de Información en su origen y desarrollo, y proyectar el análisis de la evolución histórica de los métodos de Aprendizaje de la Ordenación con la lógica de su comportamiento futuro.
  - ✓ Análisis y síntesis: A través del análisis se distinguen y revisan por separado los elementos relacionados con el Aprendizaje de la Ordenación para la R.I. Partiendo de esto, se emplea la síntesis para establecer nexos, comparar resultados, determinar características comunes y aspectos distintivos de los diferentes enfoques estudiados, lo que permite arribar a conclusiones.
  - ✓ Comparación: Para establecer las similitudes y diferencias entre los modelos y categorías existentes dentro del Aprendizaje de la Ordenación para la R.I.

1.10 Conclusiones

En el presente capítulo se presenta una revisión sintetizada del estado del arte sobre el L2R para la R.I.; 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 R.I., 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, 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 desempeño, brindan 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 más utilizadas en este campo y aquellas colecciones estándar de datos que han sido establecidas como referencia internacional.

Capítulo 2: Mejorando la tarea del L2R con FSP y reducción de datos

2.1 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 eficiente de selección de rasgos.

Finalmente se presentan las conclusiones.

2.2 Procedimiento de reducción de dimensionalidad.

2.2.1 Justificación

Microsoft Research[14] liberó el 16 junio del 2010 dos colecciones de referencia de gran tamaño para la investigación en el área del Aprendizaje de la Ordenación en RI. Estas colecciones son publicadas como: MSLR-WEB30k, que consta de más de 30 mil consultas y MSLR-WEB10K, con aproximadamente 10 mil consultas. Estas últimas constituyen un muestreo aleatorio de la primera colección. Ambos conjuntos de datos, MSLR-WEB30K y MSLR-WEB10K de tamaños 20.3GB y 6.5GB, respectivamente, resultan extremadamente grandes si se les compara con los propuestos anteriormente por Microsoft Research y Yahoo Labs[15] para asumir la tarea del L2R.

[14] Disponible en http://research.microsoft.com/en-us/projects/mslr/default.aspx
[15] Dirección web: http://labs.yahoo.com/

A pesar del inobjetable beneficio que brindarían tales colecciones a la tarea del L2R, el gran tamaño de estas puede provocar inconvenientes, tales como:

- **Un aumento considerable del tiempo de respuesta de los modelos de L2R.**

  Cuanto más consultas y documentos se tengan, 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 de los modelos sobre el conjunto de entrenamiento.**

  Al emplear un mayor número de datos, es más probable que se retengan ejemplos ruidosos, lo que provocará que esos ejemplos de escasa calidad afecten a los métodos de L2R en su generalización del modelo, y los conlleve a un sobreajuste.

Por lo tanto, si se quiere disminuir el tiempo de aprendizaje (construcción) de los modelos de L2R y alcanzar un entrenamiento escalable y eficiente, sería aconsejable reducir la dimensionalidad de los datos; manteniendo la información esencial y tratando de obtener una representación más compacta del conjunto de datos.

Esta premisa permite asegurar que el uso de estrategias y técnicas de preprocesamiento de datos facilita obtener colecciones de datos más representativas, que potencian mayores prestaciones en los modelos de L2R.

Para el preprocesado de estas colecciones de datos se utilizará la reducción de datos como estrategia. Como técnica solo nos enfocaremos en la selección de características (rasgos), debido a que en estos conjuntos tenemos un promedio de 120 documentos por consulta, donde 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, utilizar técnicas de reducción de instancias (documentos irrelevantes) a nivel de consulta ya no es oportuno porque pensamos que no resulte conveniente la selección o eliminación de documentos.

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 dataset que se estudian.

2.2.2 Selección de rasgos

Con el tiempo, el número de rasgos usados en el Aprendizaje de la Ordenación ha aumentado drásticamente. Aunque la cantidad creciente de rasgos propicia más información a los algoritmos de ranking, esta determina también mayor complejidad computacional y en cierta magnitud y para diversos casos, el sobreajuste del modelo de ranking[34]. 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 de 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.[35]

Aunque la selección de rasgos para el ranking sea vital, la mayoría de estos métodos que buscan 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, que representan una relación de ranking entre instancias; mientras que en clasificación, las categorías están "*flat*". Obviamente, existen métodos de selección de rasgos para clasificación que no resultan pertinentes para la ordenación[34].

- Segundo, las medidas de evaluación de RI (p. ej., 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)[36], mientras en la clasificación ambas precisión y exhaustividad son importantes;
    - (2) en el ranking, ordenar correctamente las primeras *n* instancias resulta más crítico[37] mientras en clasificación tomar una decisión sobre la clasificación correcta resulta de igual importancia para todas las instancias.

Frente a ambos inconvenientes, está la propuesta de desarrollar un nuevo método de selección de rasgos para la ordenación, ya que en estos últimos años se han desarrollado pocas propuestas, cuyos resultados no han logrado una buena selección de rasgos encaminados al L2R, y otras presentan inconvenientes desde su formulación.

Por ejemplo, los métodos de selección de rasgos propuestos en [34], [38], pueden manifestar un comportamiento sesgado y por tanto, resultarles 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 (p.ej., la expresión (2) en [34]), los cuales se conocen por ser no convexos y difíciles de analizar.

Las consideraciones anteriores dejan abierto un campo muy prometedor y necesario para mejorar las tareas de ranking en particular, de manera que nuestra propuesta pretende 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, los indicadores alcanzados están sustentados y determinados por sólo un grupo de consultas y no por la totalidad del conjunto.

Tal razonamiento nos puede llevar a crear un modelo de ranking que va bien para ciertas 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. A partir de esta premisa, sería conveniente considerar y aplicar 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. Una ventaja adicional de este procedimiento radicaría en que 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 relacionada 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 cuál o cuáles son los mejores y peores rasgos, es decir, aquellos rasgos que según su información, garantizan 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. 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. En tanto, el valor de ponderación negativa 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 de 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. De manera contraria se manifiesta para el caso de valores negativos.

4. Finalmente, se seleccionan de la lista ordenada los primeros ***N*** 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 cubre 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 cuáles rasgos serán seleccionados.

Ilustremos este procedimiento con un ejemplo: tomemos el conjunto de entrenamiento del Fold1 del dataset MSLR-WEB10K. En este fichero de datos, a cada fila le corresponde 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*[16].

Iniciando el procedimiento, ordenamos las instancias de la primera consulta usando los valores del rasgo con ID No.1. A partir de este ranking obtenido, evaluamos el desempeño en términos del criterio de evaluación AP, y tomamos este valor resultante como la puntuación de importancia para este rasgo. De igual manera,

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

calculamos la importancia de este rasgo para cada una de las 6000 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 la importancia de los restantes rasgos, son excluidas del procesamiento. Esta simple consideración repercute en que nos ahorremos evaluar un total de 75 mil 465 consultas para obtener el mismo resultado final.

Luego, enfocados en la primera consulta, ordenamos de forma descendente todos los rasgos acorde a sus puntuaciones y seleccionamos el mejor o los mejores rasgos (mayor valor de *AP*), así como el peor o los 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 cada uno de los rasgos. Por ejemplo, el rasgo No.1 garantiza el mejor ranking en 40 consultas y el peor desempeño en 54 consultas. En las demás consultas manifiesta un comportamiento promedio. Por tanto, el rasgo No.1 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 desempeño en la tarea del ranking para tal indicada cantidad de consultas.

A partir de estas ponderaciones finales ordenamos descendentemente, comenzando por el primero, todos los rasgos. 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.

2.3 Optimización global con FSP

2.3.1 Descripción general

El *Procedimiento de Búsqueda el Pescador* (FSP)[17] es un método de optimización global que explora nuevas soluciones combinando la búsqueda guiada y la búsqueda local. Esta metaheurística fue diseñada con el propósito de desarrollar soluciones útiles y prácticas para una variedad de problemas de optimización combinatoria.

Sus principales ventajas radican en su fácil implementación, en proporcionar una descripción explícita de la idea y constituir un 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.

2.3.2 Descripción del algoritmo

El FSP es un método de optimización global, inspirado en las tareas básicas o elementales que se ejecutan durante la pesca[39].

Para ilustrar el comportamiento de este algoritmo nos apoyaremos, mediante la figura 2.1, en la labor cotidiana de un pescador.

[17] Traducción literal del término inglés “Fisherman Search Procedure (FSP)".

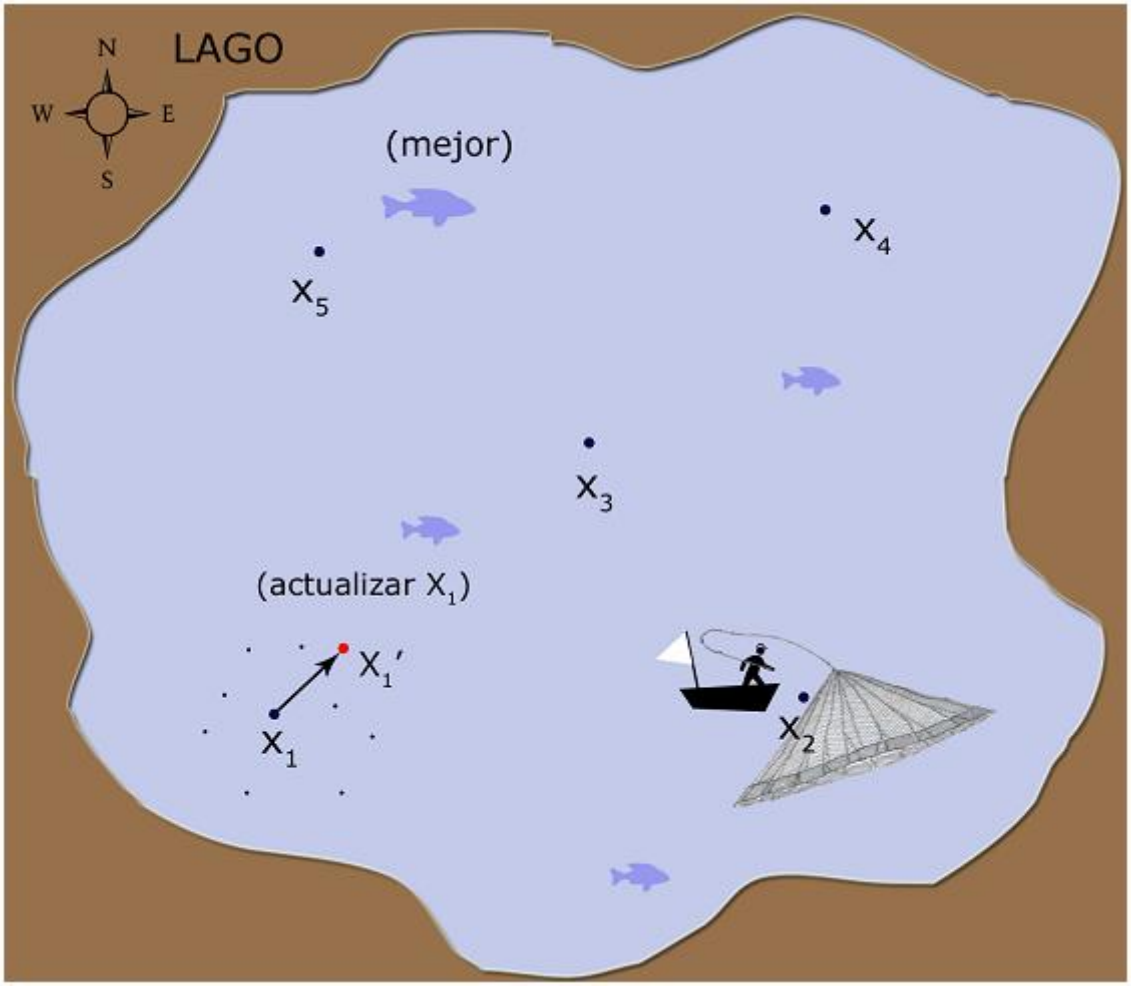


Figura 2.1: Esquema gráfico que representa el desempeño del pescador en un espacio unidimensional.[39]

Supongamos una situación real: un pescador decide ir a pescar a un lago de gran extensión y solo cuenta con una barca y varias redes de pesca. Él sabe que le resultaría imposible en un corto período de tiempo explorar toda la zona en busca de los mejores ejemplares (peces grandes). Por tanto, como todo pescador con experiencia, lo primero que hace es visualizar una serie de puntos o pequeñas áreas donde al parecer pudiese tener una buena captura. Como se muestra en la figura 2.1, tenemos cinco puntos de captura: x_1, x_2, x_3, x_4 y 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 conveniente moverse para el punto x'_1, donde vuelve a repetir la operación. Cuando en este punto sigue realizando

varios lanzamientos pero no consigue nada, se traslada hacia el siguiente punto de captura x_2 y reactiva su estrategia de pesca.

A partir de este simple procedimiento, el pescador recorre todos los puntos de captura, siempre recordando cómo se comportó la pesca en cada uno de los caladeros o pesqueros, para cuando vuelva a dichos puntos poder situarse en mejores posiciones. En toda esta actividad, según el propósito que persiga o como se vaya comportando la pesca, utiliza mallas grandes o pequeñas, y con agujeros de mayor o menor tamaño.

En la tabla 2.1 se establece el paralelo entre algunas partes del fenómeno físico de esta actividad y su aplicación a nuestro modelo de optimización.

Modelo real		Modelo de optimización
Punto de captura	⇒	Solución $x_i \in$ conjunto X de soluciones posibles
Red de pesca o malla	⇒	Número M de vectores de posición y_{ij} relativos a un x_i
Tamaño de la malla	⇒	Definido por el coeficiente de amplitud c
Lanzamientos de la malla	⇒	Cantidad L de veces que se genera y evalúa la malla en un x_i
Memoria del pescador	⇒	Por punto de captura p_i, de toda la pesca g_{best}
Rondas de pesca	⇒	Número de iteraciones T
Extensión del Lago	⇒	Espacio de búsqueda

Tabla 2.1: Equivalencias del pescador entre el modelo físico y el modelo de optimización.

Cuando extrapolamos las formas en que el pescador resuelve la pesca, se debe tener en cuenta que en el mundo real éste tiene un tiempo limitado para capturar los mejores peces. 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.

La secuencia algorítmica de FSP es mostrada en el algoritmo 2.1.

Algoritmo 2.1 : Algoritmo FSP

```
1  Input: T, N, L y M
2  for i = 1,...,N do
3  |  Inicializar aleatoriamente x_i;
4  |  Evaluar x_i;
5  |  Actualizar p_i;
6  |  Actualizar g_best;
7  end
8  for i = 1,...,T do
9  |  for j = 1,...,N do
10 |  |  while (L > 0) y (se encuentren mejoras) do
11 |  |  |  for k = 1,...,M do
12 |  |  |  |  Actualizar y_jk con la expresión (2.1);
13 |  |  |  |  Evaluar y_jk;
14 |  |  |  end
15 |  |  |  Actualizar x_j y p_j;
16 |  |  |  L = L − 1;
17 |  |  |  Aplicar estrategia para actualizar coeficiente de amplitud
18 |  |  end
19 |  |  Actualizar g_best;
20 |  end
21 end
22 Result: g_best
```

Este algoritmo en sus siete primeros pasos define un conjunto de *N* puntos de captura en toda el área 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 $X = \{x_1; x_2; \ldots; x_N\}$ 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 del paso nueve en FSP, el pescador 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).

Como haría un pescador, 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) $y_{ij} \in Y$, $Y = \{y_{i1}; y_{i2};\ldots; y_{iM}\}$. Tales vectores son 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 ($i = 1, 2,\ldots, N$; $j = 1, 2,\ldots, M$). 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.

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

$$y_{ij} = x_i + A_j \tag{2.1}$$

…donde x_i es el punto de captura desde donde se va a lanzar la red 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 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 por el estado de la pesca en cada punto de captura.

Para el paso algorítmico No.15, los vectores de posición de la red de pesca son evaluados. Y si alguno de estos logra un valor de aptitud (*fitness*) mejor que el valor p_i del punto de captura a partir del cual se lanzó la 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 referido comportamiento del pescador, cuando dentro de un área específica percibe la

necesidad de moverse hacia una mejor posición posible (como se muestra en la figura 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 también se actualiza.

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 abarcar un mayor espacio de 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 es repetida *T* veces.

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.

2.4 Método Propuesto. RankFSP

2.4.1 Formulación general

Supongamos que $Q = \{q_1, q_2, \dots, q_m\}$ es el conjunto de consultas en el entrenamiento; D constituye la colección documental y $Y = \{r_1, r_2, \dots, r_k\}$ es el conjunto de juicios de relevancia. Un juicio de relevancia constituye una medida de

cuán bueno es un documento recuperado con respecto a las necesidades de un usuario, reflejadas en la consulta formulada.

Cada consulta q_i (representada como una lista de términos $\{t_1, t_2, \dots, t_{h(q_i)}\}$ donde $h(q_i)$ denota la cantidad de términos para la i-ésima consulta) está asociada con una lista de documentos recuperados $d_i = \{d_{i_1}, d_{i_2}, \dots, d_{i_{n(q_i)}}\}$ y una lista de etiquetas $y_i = \{y_{i_1}, y_{i_2}, \dots, y_{i_{n(q_i)}}\}$, donde $n(q_i)$ denota el tamaño de la lista; $d_i \subseteq D$ e $y_i \subseteq Y$ para la consulta $q_i \in Q$, $d_{ij} \in d_i$ indica el j-ésimo documento en d_i, así como $y_{ij} \in y_i$ denota la etiqueta del documento d_{ij}. De esta manera cada documento d_{ij} está representado como una lista de rasgos o características.

Finalmente el conjunto de entrenamiento queda definido como sigue:

$$S = \{(q_i, d_i, y_i)\}_{i=1}^{m} \tag{2.2}$$

El ordenamiento se concentra en obtener una predicción para una consulta dada q_i y la lista de documentos asociados d_i, usando el modelo de ordenación.

El modelo de ordenación, denotado como f, es una función a nivel de documento, la cual constituye una combinación lineal de los rasgos en un vector de rasgos $\emptyset(q_i, d_{ij})$:

$$f(q_i, d_{ij}) = w^T \emptyset(q_i, d_{ij}) \tag{2.3}$$

Donde w denota el vector de peso y $\emptyset(q_i, d_{ij})$ es el vector de rasgos creado para cada par consulta-documento (q_i, d_{ij}), donde $i = 1, 2, \dots, m$ y j $= 1, 2, \dots, n(q_i)$. En la ordenación para la consulta q_i se asigna una puntuación para cada uno de los documentos, usando la función $f(q_i, d_{ij})$ y ordenando posteriormente tales documentos en correspondencia con las puntuaciones que les fueron dadas.

Así se obtiene una predicción denotada como π_i, que es la predicción realizada por el modelo de ordenación sobre d_i en términos de la consultaq_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, por ejemplo d_{ij}. El ordenamiento se concentraría en obtener una predicción $\pi_i \in \Pi_i$ para una consulta dada q_iy la lista de documentos asociados a d_i, usando el modelo de ordenación.

En la R.I, las medidas de desempeño son usadas para evaluar la efectividad de un modelo de ordenación, el cual usualmente está basado en consultas. Esto significa que la medida se define sobre una lista de documentos ordenados con respecto a una consulta. Entre las medidas de desempeño más empleadas están: MAP (*Mean Average Precision*)[40], NDCG (*Normalized Discounted Cumulative Gain*)[37], y P@n (*Precision @t n*)[40]. Para un análisis más detallado, consultar el apartado 1.8 Medidas de Evaluación.

En este trabajo se utiliza una función general $E(\pi_i, y_i) \in [0, +1]$ para representar las medidas de evaluación. El primer argumento de E es la predicción π_i creada usando el modelo de ordenación. El segundo argumento y_i es la lista de juicios que determina el orden ideal. E mide la correspondencia entre π_i e y_i. La mayoría de las medidas de desempeño retornan valores reales entre $[0, +1]$.

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 $E(\pi_i^*, y_i) = 1$.

En la Tabla 2.2 se presenta un resumen de las notaciones descritas anteriormente.

Notación	**Descripción**
$q_i \in Q$	Consulta
$d_i = \{d_{i_1}, d_{i_2}, \dots, d_{i_{n(q_i)}}\}$	Lista de documentos recuperados para q_i
$d_{ij} \in d_i$	*j*-ésimo documento en d_i
$y_i = \{y_{i_1}, y_{i_2}, \dots, y_{i_{n(q_i)}}\}$	Lista de juicios para d_i con respecto a q_i
$y_{ij} \in \{r_1, r_2, \dots, r_k\}$	Juicio de d_{ij} con respecto a q_i
$S = \{(q_i, d_i, y_i)\}_{i=1}^{m}$	Conjunto de entrenamiento
$\pi_i \in \Pi_i$	Predicción del modelo de ordenación para q_i
$\pi_i^* \in \Pi_i^*$	Predicción ideal para q_i

$\emptyset(q_i, d_{ij})$	Vector de rasgos para (q_i, d_{ij})
f	Modelo de ordenación
$E(\pi_i, y_i) \in [0, +1]$	Evaluación de π_i con respecto a y_i para q_i

Tabla 2.2: Resumen de notaciones para el modelo de Aprendizaje de la Ordenación.

Idealmente, el objetivo del modelo de ordenación es maximizar la precisión alcanzada sobre un conjunto de entrenamiento, en términos de una medida de desempeño de R.I.

Para ello, podemos plantear la minimización de una función de pérdida básica definida en como sigue:

$$R(f) = \sum_{i=1}^{m}\big(E(\pi_i^*, y_i) - E(\pi_i, y_i)\big) = \sum_{i=1}^{m}\big(1 - E(\pi_i, y_i)\big) \qquad (2.4)$$

…donde π_i es la predicción determinada por el modelo de ordenación para la consulta q_i.

2.4.2 Descripción general

El método RankFSP es una variante adaptada y mejorada del Procedimiento de Búsqueda del Pescador para tratar el problema del L2R, capaz de construir una función de ordenación optimizando directamente cualquier medida de evaluación según un determinado conjunto de entrenamiento. Este nuevo método 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 grandes colecciones de datos y específicamente sobre las que fueron analizadas en el apartado 1.7 del Capítulo 1 de esta tesis de investigación.

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 fueron incorporadas a este algoritmo.

2.4.3 Estrategias incorporadas

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 determinada 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}.

A diferencia de FSP, donde en cada punto de captura el pescador siempre realiza una cantidad fija de lanzamientos, en RankFSP esta cantidad de tiros de la malla está condicionada a mejorar el valor de la función en p_i. Es decir, mientras que con cada lanzamiento se vaya mejorando la solución actual para un punto de captura, el pescador continúa lanzando la malla hasta que no encuentre mejoras. En ese caso, se mueve para el siguiente punto de captura.

RankFSP también incorpora una estrategia para evadir los estancamientos en mínimos locales o en zonas donde no se encuentren mejoras. Para ello, después de un rango de iteraciones, identifica aquellos puntos de captura cuya posición no fue mejorada y donde no encontró ninguna buena solución a partir de ellos, y los redefine a nuevas posiciones de pesca. Este reinicio puede ser aleatorio o controlado (guiado hacia zonas de pesca inexploradas). Estas variantes y aportes de RankFSP serán contextualizadas con la propia descripción del algoritmo.

2.4.4 Descripción del algoritmo

El procedimiento general de RankFSP es mostrado en el Algoritmo 2.2. Los parámetros de entrada de RankFSP son: 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 siete pasos del algoritmo, RankFSP al igual que FSP, inicializa aleatoriamente en 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).

Algoritmo 2.2 : Algoritmo RankFSP

1 **Input**: $S = \{(q_i, d_i, y_i)\}_{i=1}^{m}$, E, T, N y M
2 **for** i = *1*,...,N **do**
3 Inicializar aleatoriamente x_i;
4 Evaluar x_i;
5 Actualizar p_i;
6 Actualizar g_{best};
7 **end**
8 **for** k = *1*,...,T **do**
9 **for** i = *1*,...,N **do**
10 **while** *(se encuentren mejoras)* **do**
11 **for** j = *1*,...,M **do**
12 Actualizar y_{ij} con la expresión (2.1): $y_{ij} = x_i + A_j$;
13 Evaluar y_{ij} con la expresión (2.4): $R(f) = \sum_{i=1}^{m} (1 - E(\pi_i, y_i))$;
14 **end**
15 Actualizar x_i y p_i;
16 Aplicar estrategia para actualizar coeficiente de amplitud
17 **end**
18 Actualizar g_{best};
19 **end**
20 Cambiar la posición de los puntos de captura no mejorados
21 **end**
22 Construir la función de ranking f con el vector de posición g_{best};
23 **Result**: f

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, el pescador se mantendrá en ese lugar 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.

Siguiendo el paso No.13 del algoritmo, 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 ecuación $y_{ij} = x_i + A_j$ 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 $\pi_{i,}$ obtenida a partir de la aplicación de la expresión $f(q_i, d_{ij}) = w^T \emptyset(q_i, d_{ij})$ (2.3) 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 global g_{best}.

Durante la pesca, nuestro algoritmo implementa una estrategia 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* (p.ej., 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 muerto) y, por lo tanto, procedemos a redefinir 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 nuevas o no exploradas zonas de pesca. 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, en el paso No.22 el modelo de ranking f es construido con el vector de posición (*gbest*) obtenido en la última iteración.

2.5 Conclusiones

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).

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

En esencia, aunque cada uno de estos métodos presenta ventajas y desventajas entre sí, lo más importante es que toman partida de su concepción y sus estrategias para acometer la tarea del L2R de la mejor manera.

Capítulo 3: Resultados experimentales

3.1 Introducción

En este capítulo, 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.

3.2 Desempeños obtenidos

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 2500 a 5000 evaluaciones para las colecciones utilizadas. La medida de evaluación que utilizamos en la ecuación (2.3) 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[18] 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 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 2.2.2. 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 modelos 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.

[18] Dirección web http://research.microsoft.com/en-us/um/beijing/projects/letor/

2. Eliminar las consultas consideradas como *outliers*[19], es decir, aquellas 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[20] 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 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.

3.2.1 Comparación con métodos referenciados

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.

[19] 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.

[20] 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/).

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.

Por otro lado, a diferencia de lo que ocurre con otras colecciones archiconocidas como OHSUMED, Gov y Gov2; donde los resultados experimentales de 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 publicados [41], [42] 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 (p. ej., AdaRank-MAP[32], RankingSVM(RSVM)[21][43]), otros emplean una función de ranking no lineal (p. ej., RankBoost[26]) y otros llevan a cabo la tarea del L2R en lnea (online) como: PARank-NDCG[42], Committee Perceptron (ComP)[44] para este caso implementado en C++ por Suhara en [42] y Stochastic Pairwise Descent (SPD)[45] implementado con sofia-ml[22]. Para este último algoritmo, se emplearon dos versiones utilizando Pegasos-SVM (SPD-SVM) y Passive-Aggressive (SPD-PA) como métodos de aprendizaje ya incorporados en el paquete de sofia-ml.

La Tabla 3.1 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

[21] Usando svm_rank como implementación disponible en URL: http://www.cs.cornell.edu/people/tj/svmlight/svm rank.html

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

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 y RankPSO, obteniendo RankFSP+Prep el mejor resultado en el ranking. Este comportamiento se manifiesta de igual forma en los resultados obtenidos por RankFSP y RankPSO frente a MSLR-WEB30K, cuyos valores pueden ser consultados en la parte (b) de dicha tabla.

(a) MSLR-WEB10K

Algoritmo	MAP	Breve descripción
BM25	0.2561	*Single feature ranker* [35]
RankBoost	0.2770	*Boosting algorithm* que minimiza la discordancia a pares en el *ranking* [19]
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[5]
RankFSP	0.3044	Algoritmo de optimización directa basado en FSP
AFS	0.3053	Co-ordinate ascent-based algorithm with direct optimization for MAP [133]
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
RankPSO	0.2952	Algoritmo de optimización directa inspirado en PSO [100]
RankFSP	0.3021	Algoritmo de optimización directa basado en FSP
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 3.1: Rendimientos alcanzados por algoritmos de L2R sobre los datasets MSLR-WEB10K (a) y MSLR-WEB30K (b), considerando MAP como medida de evaluación.

Entretanto, la Tabla 3.2 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 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.

Es notorio mencionar además que nuestro método RankFSP+Prep alcanza tales resultados con un valor máximo de 5000 evaluaciones completas. Pues según [42], 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.

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

Por otra parte, en la Tabla 3.3 solo incluimos los resultados de precisión de RankFSP y RankPSO, con dos variantes de presentación, es decir, con y sin preprocesamiento de los conjuntos de entrenamiento. El resto de los métodos de 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 dataset.

(a) MSLR-WEB10K

Algoritmo	P@1	P@2	P@3	P@4	P@5	P@6	P@7	P@8	P@9	P@10
RankPSO	0.4030	0.2132	0.1347	0.0890	0.0858	0.0642	0.0620	0.0602	0.0583	0.0523
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
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	0.0797	0.0699	0.0622	0.0589	0.0518
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 3.3: 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 datasets y medidas 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.

3.2.2 Análisis estadístico

Después de haber obtenido y comparado los resultados de rendimiento de nuestro método propuesto, 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 y RankPSO, incluyendo sus aplicaciones posteriores a la selección de rasgos, 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) obtenidos 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 10000 valores de AP por cada algoritmo, y para MSLR-WEB30K un total de 31531.

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)[46] de dos vías de Friedman por rangos[47], [48], incluyendo las comparaciones múltiples por parejas, y usando un intervalo de confianza del 95%.

En la figura 3.1 se muestran cómo 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	26.845,708
Grados de libertad	3
Sig. asintótica (prueba de dos caras)	,000

(a) MSLR-WEB10K

N	31.531
Chi-cuadrado	83.627,944
Grados de libertad	3
Sig. asintótica (prueba de dos caras)	,000

(b) MSLR-WEB30K

Figura 3.1: Estadísticos de contraste obtenidos para ambas colecciones de datos.

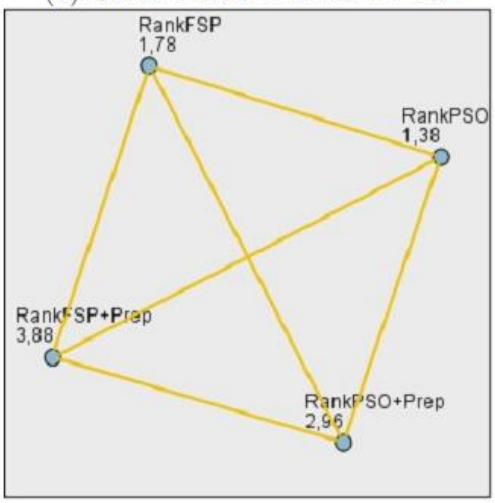
Entretanto, para el caso del dataset MSLR-WEB10K la figura 3.2 visualiza como se desarrollaron las comparaciones por parejas a través de un gráfico de distancias de red (parte a) y una tabla de comparaciones (parte b). En el gráfico las distancias entre nodos de la red corresponden a las diferencias entre las muestras, y como estas son de color amarillo[23] podemos decir que existen

[23] En estos gráficos de SPSS las líneas de color amarillo entre nodos representan diferencias significativas, mientras las de color negro indican que las diferencias no son notables.

diferencias estadísticas importantes entre todos los algoritmos.

El valor de las etiquetas en los nodos se corresponde 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.

(a) Gráfico de distancias de red



(b) Tabla de comparaciones

Muestra1-Muestra2	Prueba estadística	Error típico	Desv. Prueba estadística	Sig.	Sig. adv.
RankPSO-RankFSP	-,400	,018	-21,936	,000	,000
RankPSO-RankPSO+Prep	-1,578	,018	-86,455	,000	,000
RankPSO-RankFSP+Prep	-2,497	,018	-136,780	,000	,000
RankFSP-RankPSO+Prep	-1,178	,018	-64,519	,000	,000
RankFSP-RankFSP+Prep	-2,097	,018	-114,844	,000	,000
RankPSO+Prep-RankFSP+Prep	-,919	,018	-50,325	,000	,000

Figura 3.2: Resumen de las comparaciones múltiples por parejas de los algoritmos en MSLR-WEB10K.

En este sentido, podemos percibir cómo para todos los casos comparados, los métodos RankFSP+Prep y RankPSO+Prep mejoran con un alto nivel de

significancia estadística a RankFSP y RankPSO, lo que reafirma los beneficios del método propuesto de selección de rasgos. Además, se observa como RankFSP tiene diferencias significativas pero en un menor grado con respecto a RankPSO, y del mismo modo RankFSP+Prep refleja sus diferencias con RankPSO+Prep.

Para el caso del dataset MSLR-WEB30K las diferencias estadísticas entre los algoritmos se dan de igual forma que en MSLR-WEB10K. Para más detalles sobre los resultados en MSLR-WEB30K, consultar la figura del Anexo 2.

3.2.3 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 3.2 (Desempeños obtenidos), 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 5000 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 3.4 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 3.4: 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 3.5: Tiempo de procesamiento (horas) para construir una función de ranking a partir de cada conjunto de entrenamiento.

Siguiendo está lógica, en la tabla 3.5 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 3.4 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 No.1 la totalidad de las consultas, es decir, 6000; 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). Como 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 consultas24.

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 3.6, donde se especifica además, la cantidad de consultas que nos evitamos evaluar utilizando los procedimientos de selección de rasgos y filtrado.

[24] Valor resultante de la expresion (6000-742)*5000

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 3.6: 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 3.2, 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.

3.3 Conclusiones

En el presente capítulo se realizó un estudio experimental para evaluar el desempeño del método propuesto RankFSP, considerando para ello el rendimiento alcanzado en la ordenación para los dos conjuntos de datos MSLR-WEB10K y MSLR-WEB30K. Los resultados obtenidos para ambas variantes de RankFSP y RankPSO 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 adicionalmente el tiempo de entrenamiento y la cantidad de consultas a evaluar.

Conclusiones generales

1. El análisis de las tendencias categóricas en L2R y de los modelos más representativos, sugiere 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.

2. Con el objetivo de mejorar la tarea del ranking se propuso 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.

3. 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 alcanzaron los mejores valores de precisión en el ranking en comparación con los demás algoritmos estudiados, y en relación a las colecciones de gran tamaño.

4. Los resultados que mostraron RankFSP y RankPSO 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.

Recomendaciones

- Conducir más experimentos con nuevas colecciones de datos con el fin de verificar las potencialidades y rendimientos de RankFSP.

- Probar la efectividad de la propuesta de selección de rasgos en nuevas y referenciadas colecciones de datos.

- 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.

Referencias Bibliográficas

[1] G. Salton y M. J. McGill, *Introduction to Modern Information Retrieval*. New York: McGraw–Hill, 1983.

[2] C. N. Mooers, «The Theory of Digital Handling of Non-Numerical Information and its Implications to Machine Economics», *Zator Technical Bulletin #48*, nº. 48, pág. 34, 1950.

[3] C. T. Meadow, *Text Information Retrieval Systems*, Tercera Edición. 84 Theobald's Road, London: , 2007.

[4] D. A. Grossman, *Information Retrieval: Algorithms and Heuristics*, Ilustrada. Netherland: Springer, 2004.

[5] J. Perez-Carballo y T. Strzalkowski, «Information Processing and Management 36 p.155-178», in *Natural language information retrieval: progress report*, 2000.

[6] C. van Rijsbergen, *Information Retrieval*. Butterworth-Heinemann, 1979.

[7] G. Salton, E. A. Fox, y H. Wu, «Extended Boolean information retrieval», *Magazine Communications of the ACM*, vol. 26, nº. 11, págs. 1022-1036, 1983.

[8] G. Salton, A. Wong, y C. S. Yang, «A vector space model for automatic indexing», *Magazine Communications of the ACM*, vol. 18, nº. 11, págs. 613-620, 1975.

[9] W. B. Croft y D. J. Harper, «Using probabilistic models of document retrievalwithout relevance information», vol. 35, nº. 4, págs. 285-295, 1979.

[10] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, y M. Gatford, «Okapi at TREC-3», presented at the Third Text REtrieval Conference (TREC-3), págs. 109-126.

[11] C. D. Manning y H. Schütze, *Foundations of Statistical Natural Language Processing.* 1999.

[12] R. Rosenfeld, «Two decades of statistical language modeling: Where do we go fromhere?», presented at the IEEE, 2000, págs. 1270-1278.

[13] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, y R. A. Harshman, «Indexing by Latent Semantic Analysis», *American Society for Information Science*, vol. 41, nº. 6, págs. 391-407, 1990.

[14] D. D. Lewis y K. Spärck Jones, «Natural language processing for information retrieval», presented at the CACM, 1996, vol. 39, págs. 92-101.

[15] J. M. Ponte y W. B. Croft, «A language modeling approach to information retrieval», presented at the 21st annual international ACM SIGIR conference on Research and development in information retrieval, 1998, págs. 275-281.

[16] C. Zhai y J. Laff_erty, «A study of smoothing methods for language models applied to Ad Hoc information retrieval», presented at the 24th annual international ACM SIGIR conference on Research and development in information retrieval, 2001, págs. 334-342.

[17] S. Brin y L. Page, «The Anatomy of a Large-Scale Hypertextual Web Search Engine», *Journal Computer Networks and ISDN Systems*, vol. 30, nº. 1-7, págs. 107-117, 1998.

[18] Z. Gyongyi, H. Garcia-Molina, y J. Pedersen, «Combating web spam with TrustRank», presented at the Thirtieth international conference on Very large data bases, 2004, vol. 30, págs. 576-587.
[19] Y. Liu et al., «BrowseRank: letting web users vote for page importance», presented at the 31st annual international ACM SIGIR conference on Research and development in information retrieval, 2008, págs. 451-458.
[20] F. S. Briceño Segovia, «Clasificación Automática de textos basado en Ranking», Universidad Católica de Valparaíso, 2011.
[21] T.-Y. Liu, *Learning to rank for information retrieval*, vol. 3. Microsoft Research Asia, 2007.
[22] S. Vargas Sandoval, «Learning to Rank: Técnicas de Aprendizaje Automático en Recuperación de Información», Universidad Autónoma de Madrid, Madrid, España, 2001.
[23] K. Crammer y Y. Singer, «Pranking with ranking», presented at the 15th Annual Conference on Neural Information Processing Systems., NIPS 2001.
[24] C. J. C. Burges, «Learning to rank using gradient descent», presented at the 22nd interna tional conference on Machine learning, ICML 2005, New York, USA.
[25] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, y W.-Y. Ma, «Frank: a ranking method with fidelity loss», presented at the 30th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR 2007, New York, USA.
[26] Y. Freund, R. Iyer, R. E. Schapire, y Y. Singer, *An efficient boosting algorithm for combining references. Journal of Machine Learning Research*. 2003.
[27] R. Herbrich, T. Graepel, y K. Obermayer, «Large margin rank boundaries for ordinal regression», *MIT Press*, Cambridge, págs. 115–132, 2000.
[28] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, y H. Li, «Learning to rank: from pairwise approach to listwise approach», presented at the 24th international conference on Machine learning, ICML 2007, New York, USA, págs. 129–136.
[29] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, y H. Li, «Listwise approach to learning to rank theory and algorithm», presented at the 25th international conference on Machine learning, ICML 2008, New York, USA.
[30] M. Taylor, J. Guiver, S. Robertson, y T. Minka, «Softrank: optimizing nonsmooth rank metrics», presented at the international conference on Web search and web data mining, New York, USA, 2008, págs. 77–86.
[31] Y. Yue, T. Finley, F. Radlinski, y T. Joachims, «A support vector method for optimizing average precision», presented at the 30th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR 2007, New York, USA, 2007.
[32] J. Xu y H. Li, «Adarank: a boosting algorithm for information retrieval», presented at the 30th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR 2007, New York, USA, págs. 391–398.
[33] S. E. Robertson y K. S. Jones, «Relevance weighting of search terms», *Journal of the American Society for Information Science*, vol. 27, nº. 3, págs. 129-146, 1976.

[34] R. A. Baeza-Yates y B. Ribeiro-Neto, *Modern information retrieval*, First Edition. 1999.
[35] F. Pan, T. Converse, D. Ahn, F. Salvetti, y G. Donato, «Feature selection for ranking using boosted trees», presented at the 18th ACM conference on Information and knowledge management, New York, USA, 2009, págs. 2025-2028.
[36] V. Dang y W. B. Croft, «Feature Selection for Document Ranking using Best First Search and Coordinate Ascent», presented at the SIGIR Workshop on Feature Generation and Selection for Information Retrieval, New York, USA, 2010.
[37] K. Järvelin y J. Kekäläinen, «Cumulated Gain-Based Evaluation of IR Techniques», presented at the ACM Transactions on Information Systems, 2002, págs. 422–446.
[38] P. Gupta y P. Rosso, «Expected Divergence based Feature Selection for Learning to Rank», presented at the 24th International Conference on Computational Linguistics, Mumbai, India, 2012, págs. 431-440.
[39] O. J. Alejo Machado, J. M. Fernández-Luna, J. F. Huete-Guadix, y E. Concepción-Morales, «Fisherman Search Procedure. Progress in Artificial Intelligence. Special Issue: Metaheuristics Applied To Solve Real World Problems», 2014.
[40] T.-Y. Liu, T. Qin, W.-Y. Xiong, H. Li, y J. Xu, «Letor: Benchmark dataset for research on learning to rank for information retrieval», presented at the SIGIR 2007 Workshop on Learning to Rank for Information Retrieval.
[41] A. Lumini y L. Nanni, «A clustering method for automatic biometric template selection», *Pattern Recognition*, vol. 39, nº. 3, págs. 495-497, 2006.
[42] J. A. Olvera-López, J. A. Carrasco-Ochoa, y J. F. Martínez-Trinidad, «Prototype Selection Via Prototype Relevance», *Pattern Recognition: Image Analysis and Applications*, vol. 5197, págs. 153-160, 2008.
[43] T. Joachims, «Optimizing Search Engines Using Clickthrough Data», presented at the 8th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, USA, 2002, págs. 133-142.
[44] P. Hart, «The condensed nearest neighbor rule», presented at the IEEE Transactions on Information Theory, 1968, vol. 14, págs. 515-516.
[45] H. Brighton y C. Mellish, «Advances in instance selection for instance-based learning algorithms», *Data Mining and Knowledge Discovery*, vol. 6, nº. 2, págs. 153-172, 2002.
[46] D. J. Sheskin, *Handbook of Parametric and Nonparameteric Statistical Procedures*, 3o ed. 2003.
[47] M. Friedman, «A comparison of alternative tests of significance for the problem of m rankings», *Annals of Mathematical Statistics*, vol. 11, nº. 1, págs. 86-92, 1940.
[48] C.-F. Tsai y C.-W. Chang, «SVOIS: Support Vector Oriented Instance Selection for text classification», *Information Systems*, vol. 38, nº. 8, págs. 1070-1083, 2013.

Anexos

Anexo 1: Lista de funciones de Microsoft Learning para clasificar los conjuntos de datos

Id de característica	Descripción de la característica	Flujo	Comentarios
1	consulta cubierta término número	cuerpo	
2		ancla	
3		Título	
4		URL	
5		todo documento	
6	cociente del término consulta cubierto	cuerpo	
7		ancla	
8		Título	
9		URL	
10		todo documento	
11	longitud de la secuencia	cuerpo	
12		ancla	
13		Título	
14		URL	
15		todo documento	
16	FDI (frecuencia documento inverso)	cuerpo	
17		ancla	
18		Título	
19		URL	
20		todo documento	
21	suma de la frecuencia del término	cuerpo	
22		ancla	
23		Título	
24		URL	
25		todo documento	
26	min de frecuencia del término	cuerpo	
27		ancla	
28		Título	
29		URL	
30		todo documento	
31	máximo de frecuencia de término	cuerpo	
32		ancla	
33		Título	
34		URL	
35		todo documento	

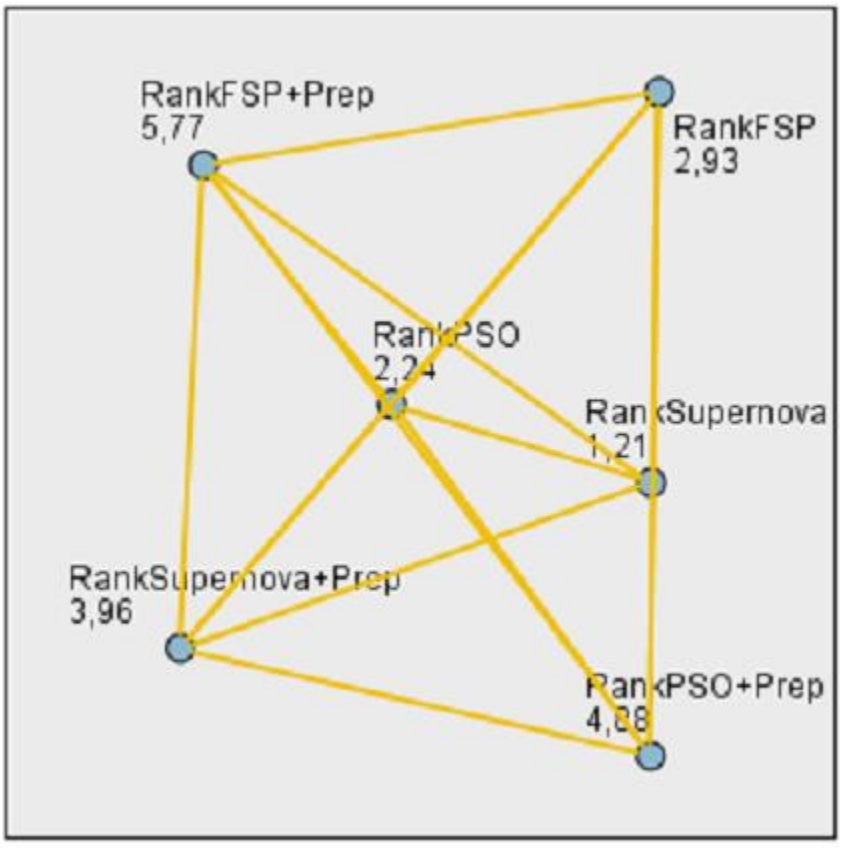
36	media de frecuencia de término	cuerpo	
37		ancla	
38		Título	
39		URL	
40		todo documento	
41	variación de la frecuencia del término	cuerpo	
42		ancla	
43		Título	
44		URL	
45		todo documento	
46	suma de corriente longitud normalizada término frecuencia	cuerpo	
47		ancla	
48		Título	
49		URL	
50		todo documento	
51	min de flujo longitud normalizada término frecuencia	cuerpo	
52		ancla	
53		Título	
54		URL	
55		todo documento	
56	máximo de corriente longitud normalizada término frecuencia	cuerpo	
57		ancla	
58		Título	
59		URL	
60		todo documento	
61	media de corriente longitud normalizada término frecuencia	cuerpo	
62		ancla	
63		Título	
64		URL	
65		todo documento	
66	variación de corriente longitud normalizada término frecuencia	cuerpo	
67		ancla	
68		Título	
69		URL	
70		todo documento	
71	suma de tf * idf	cuerpo	
72		ancla	
73		Título	
74		URL	
75		todo documento	
76	min de tf * idf	cuerpo	
77		ancla	
78		Título	
79		URL	
80		todo documento	

81	máximo de tf * idf	cuerpo	
82		ancla	
83		Título	
84		URL	
85		todo documento	
86	promedio de tf * idf	cuerpo	
87		ancla	
88		Título	
89		URL	
90		todo documento	
91	varianza de tf * idf	cuerpo	
92		ancla	
93		Título	
94		URL	
95		todo documento	
96	modelo booleano	cuerpo	
97		ancla	
98		Título	
99		URL	
100		todo documento	
101	modelo de espacio vectorial	cuerpo	
102		ancla	
103		Título	
104		URL	
105		todo documento	
106	BM25	cuerpo	
107		ancla	
108		Título	
109		URL	
110		todo documento	
111	LMIR.ABS	cuerpo	Enfoque del modelo de lengua para recuperación de información (IR) con absoluta descuento alisado
112		ancla	
113		Título	
114		URL	
115		todo documento	
116	LMIR.DIR	cuerpo	Enfoque del modelo de lengua para IR con Bayesiano alisar utilizando antecedentes de Dirichlet
117		ancla	
118		Título	
119		URL	
120		todo documento	

121	LMIR.JM	cuerpo	Enfoque del modelo de lengua para IR con suavizado Jelinek-Mercer
122		ancla	
123		Título	
124		URL	
125		todo documento	
126	Número de slash en URL		
127	Longitud de la URL		
128	Número de vínculo		
129	Número de outlink		
130	PageRank		
131	SiteRank		Nivel de sitio PageRank
132	QualityScore		La puntuación de calidad de una página web. La puntuación se genera por un clasificador de calidad de página web.
133	QualityScore2		La puntuación de calidad de una página web. La puntuación se genera por un clasificador de calidad de página web, que mide la maldad de una página web.
134	Consulta-url, haga clic en cuenta		El Conde de clic de un par de consulta-url en un motor de búsqueda en un período de
135	URL, haga clic en cuenta		El Conde de clic de una dirección url acumulados de datos en un periodo de navegación del usuario
136	tiempo de permanencia de URL		El tiempo promedio de una

			dirección url acumulados de datos en un periodo de navegación del usuario

Anexo 2: Resumen de las comparaciones múltiples por parejas de los algoritmos en MSLR-WEB30K.



(a) Gráfico de distancias de red.

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

(b) Tabla de comparaciones.