UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

Metabuscador con gestión de sesión

Javier Sanz-Cruzado Puig

Tutor: Pablo Castells Azpilicueta

JUNIO 2015

Resumen

Los buscadores son hoy en día la herramienta más ampliamente usada en la Web. Este hecho ha propiciado que las tecnologías, métodos y algoritmos de recuperación de información hayan experimentado una gran desarrollo y una rápida evolución, dando lugar a funcionalidades sofisticadas, así como perspectivas complejas y matizadas sobre la calidad y utilidad de la salida de un buscador. Este desarrollo es imposible sin el desarrollo paralelo de metodologías adecuadas para evaluar y constrastar diferentes soluciones y sistemas, que permitan en definitiva determinar, medir y comparar lo buena que es la respuesta que se proporciona al usuario. Esto ha tenido como consecuencia la concepción de nuevos métodos para realizar la evaluación y comparativa de diferentes sistemas de recuperación de información de información, incluyendo los sistemas de búsqueda en la web.

En el presente trabajo se ha desarrollado una aplicación que persigue un doble objetivo. En primer lugar, ofrecer funcionalidades avanzadas de búsqueda sobre un metabuscador que combine resultados de diversos buscadores comerciales. En concreto, se han implementado técnicas y algoritmos de diversificación de resultados, *relevance feedback* y gestión de sesiones de búsqueda, sobre una agregación de los buscadores Google, Bing, Carrot y Faroo. La diversificación de resultados permite espaciar entre sí los documentos similares en el ranking de resultados. Por otro, las técnicas de relevance feedback, que añaden palabras a una consulta en función de las acciones previas de un usuario en una sesión de búsqueda (un grupo de consultas orientadas a satisfacer una única necesidad de información). Por último, se han desarrollado métodos para realizar la gestión de dichas sesiones a partir de las consultas realizadas por los usuarios, tanto de manera implícita (gestión interna), como de manera explícita, mediante un sistema que permite a los usuarios guardar aquellos documentos que les parezcan más interesantes, así como crear y guardar sesiones de búsqueda.

En segundo, se ha implementado en el presente trabajo un sistema que permite la evaluación simultánea de diferentes buscadores web, a partir de métodos de intercalado de resultados. Para alcanzar este objetivo, se ha desarrollado un método probabilístico de intercalado de resultados que permite realizar la evaluación de diversos buscadores a partir de las interacciones de los usuarios con el sistema, sin que el usuario observe ninguna diferencia en el aspecto de la aplicación respecto al uso normal.

Palabras clave: recuperación de información, búsqueda, metabúsqueda, sesión de búsqueda, relevance feedback, diversidad, evaluación, test A/B, test multivariante

Abstract

Search engines are today the most widely and frequently used applications on the Web. That fact has favoured a great development and a fast evolution of information retrieval technologies, methods and algorithms, giving room to sophisticaded functionalities, as well as complex perspectives about the utility and quality of a searcher output. This development is impossible without the parallel development of adequate technologies for evaluating and contrast different solutions and systems, in order to determine, measure and compare how good the provided response to the user is. This has caused the conception of new methods for evaluating and comparing information retrieval systems, including search ones.

In this thesis, a web application has been developed, which follows a dual objective. First, offering advance search functionalities over a metasearcher that combines results from several commercial search engines. Specifically, methods and algorithms of result diversification, relevance feedback and management of search sessions have been implemented over an aggregation of the searchers Google, Bing, Carrot and Faroo. Result diversification methods allow the spacing of similar documents from each other within a raking of search results. Furthermore, relevance feedback techniques add words to a given query based on the previous actions of the user in a search session (a group of queries aimed at satisfying a single information need). Finally, methods for managing that sessions have been implemented. Two approaches have been developed: In one hand, an internal session identification and management. On the other hand, an explicit session management, that allows users to store those documents more interesting to them, as well as creating and saving search sessions.

Secondly, in the present thesis, a system that allows the simultaneous evaluation of several web search engines, via the interleaving of results has been developed. To achieve this goal, a probabilistic interleaving method that permits the evaluation of multiple search engines by analizing the interactions of the users with the application has been developed, without the user noticing any difference in the application appeareance in relation to the normal use.

Keywords: information retrieval, search, metasearch, search session, relevance feedback, diversity, evaluation, A/B test, multivariate test

Agradecimientos

Quiero dar las gracias a mi familia, por todo su apoyo y cariño durante todos estos años. También quiero dar las gracias a todos mis compañeros de carrera, en especial a Rubén, mi compañero de prácticas durante casi toda la carrera,y a Rus y Laura, por los infinitos mensajes de ayuda para poder resolver ejercicios de matemáticas. Sin todos vosotros, no podría haber llegado tan lejos.

Me gustaría dar las gracias también a todos aquellos que han colaborado en este trabajo con sus búsquedas, y, por último, agradecer a mi tutor, Pablo, por la oportunidad de realizar este trabajo, y por toda la ayuda recibida durante estos meses.

Índice

Índice de tablas

Índice de figuras

Glosario

Buscador: Sistema que recupera documentos en un espacio de información masivo y/o no estructurado a partir de una consulta.

Consulta: Formulación, por parte de un usuario, de una necesidad de información por medio de unas pocas palabras.

Diversificación de resultados: Conjunto de técnicas orientadas a maximizar la probabilidad de que al menos un documento del ránking sea escogido por el usuario, reordenando un ránking para no situar en posiciones próximas documentos muy similares, o que traten del mismo tema.

Metabuscador: Buscador que combina los documentos recuperados por otros buscadores en un único ranking unificado.

Metasesión: En este trabajo, sesión de búsqueda gestionada por el usuario, en la que se permite el guardado de documentos relevantes.

Necesidad de información: Información de la que el usuario precisa para realizar un objetivo.

Pseudo-relevance feedback: Variante de relevance feedback que considera parte de los documentos recuperados por el sistema como relevantes, y añade palabras a la consulta en función del contenido de dichos documentos, independientemente de las acciones de los usuarios. También conocido como blind relevance feedback.

Ranking: Lista ordenada de resultados de búsqueda.

Relevancia: Propiedad que cumple un documento que satisface una necesidad de información para la que se ha formulado una consulta.

Relevance feedback: Conjunto de técnicas que reformulan una determinada consulta añadiendo nuevas palabras en función de las interacciones del usuario que realizó la consulta en una sesión de búsqueda.

Score: Puntuación obtenida por un documento web en un determinado buscador para conformar el ranking.

Sesión de búsqueda: Conjunto de consultas realizadas por un usuario orientadas a satisfacer una única necesidad de información.

Similitud: Medida empleada para determinar lo parecidos que son dos documentos.

Snippet: Breve resumen de un documento web, devuelto habitualmente por los buscadores.

Test A/B: Método de evaluación que permite evaluar dos sistemas, o dos variantes de un sistema, a partir de las interacciones de los usuarios con los mismos.

Test multivariante: Generalización de los Test A/B para evaluar un número indefinido de sistemas.

1. Introducción

1.1 Motivacion

Desde los años 50 del siglo pasado, la recuperación de información ha desarrollado métodos, teorías, tecnologías y soluciones a diferentes problemas del ámbito de acceso a la información en espacios masivos, tales como la búsqueda, la clasificación o la detección de topics. Si bien el fundamento de la mayoría de las técnicas se concibió durante los años 60 y 70, es desde los años 90, con la aparición de la Web, que estas se han expandido enormemente.

En particular, dado el enorme tamaño de la misma (al menos, miles de millones de documentos), los sistemas de búsqueda han experimentado una gran evolución. En las últimas dos décadas, los buscadores *online* se han convertido en una de las herramientas más populares disponibles en la red, y algunos, como Google, Bing, Baidu o Yandex son usados cada día por cientos de millones de personas para tratar de satisfacer sus necesidades de información.

El desarrollo de las tecnologías de recuperación de información no es posible sin una adecuada evaluación de las distintas soluciones y sistemas, que determine lo buena que es la respuesta que se le proporciona al usuario. Esta evaluación ha de ser siempre comparativa entre diferentes soluciones o alternativas, y se basa en determinar qué documentos entre los devueltos por el sistema satisfacen la necesidad de los usuarios que formulan las consultas. La forma tradicional de medir la utilidad de un documento ha girado en torno al concepto de relevancia, si bien desde hace algo más de una década se ha empezado a matizar este concepto con otras consideraciones claves tales como la no redundancia, la multidimensionalidad de la utilidad de los documentos, el contexto de las búsqueda , o los diferentes sentidos o interpretaciones de una misma consulta.

Tanto el desarrollo de soluciones y algoritmia de recuperación de información como el diseño de métodos de evaluación han evolucionado enormemente a lo largo de las últimas décadas. Con respecto a los motores de búsqueda, las innovaciones más recientes tienden a la personalización de las búsquedas de cada usuario, teniendo en cuenta las consultas anteriores. Se han desarrollado técnicas como el *relevance feedback*, que trata de mejorar los resultados obtenidos añadiendo palabras al final de la consulta, teniendo en cuenta las acciones anteriores (clicks y consultas) del usuario en una sesión de búsqueda.

Aparece la sesión de búsqueda como noción de que las consultas individuales no son acciones aisladas sino que en muchos casos tienen una relación entre sí que el sistema puede tener en cuenta para mejorar la calidad de sus respuestas. Las sesiones de búsqueda constan de un grupo de consultas destinadas a satisfacer una necesidad de información (Jansen et al. 2007), e introducen la oportunidad de analizar el comportamiento del usuario, para asistirle mejor en satisfacer sus objetivos.

Más allá de la búsqueda básica y las nociones de calidad basadas en relevancia, aparecen dos nuevos paradigmas, la metabúsqueda y la diversidad. La metabúsqueda es una técnica que permite incorporar diferentes motores de búsqueda en un único ranking (ejemplos de ello son buscadores como iZito o eTools). La diversidad, por otro lado, como técnica, reordena los resultados de búsqueda procurando distanciar en el ranking los documentos similares entre sí. Esta técnica es empleada de uno u otro modo, y hasta cierta medida, por la mayoría de los buscadores comerciales. Además de ello, la diversidad también se puede tener en cuenta al evaluar sistemas de recuperación de información, estudiando la similitud entre los diferentes elementos que conforman el ranking.

La evaluacíon es una de las áreas más amplias y troncales del campo de la recuperación de información, a la que se ha dedicado una porción muy significativa de la fuerza de trabajo de la

comunidad académica y comercial. Entre los múltiples paradigmas y métodos de evaluación, en este TFG nos centramos en la llamada evaluación online, que se puede realizar sobre un sistema en vivo (por contraste con los experimentos en laboratorio), de forma que esta evaluación sea transparente para los usuarios, y más concretamente en los tests de tipo A/B basados en los clicks de los usuarios y extensibles a la comparación simultánea de un número arbitrario de motores de búsqueda.

El presente trabajo aborda las anteriores ideas para diseñar una aplicación web que permita, por un lado, aplicar técnicas avanzadas de búsqueda sobre los resultados obtenidos de varios buscadores, y por otro lado, evaluar esos u otros buscadores simultáneamente. Es decir, el presente trabajo tiene una doble motivación: por un lado, presentar una plataforma que permita combinar diferentes buscadores web , y evaluarlos comparativamente, y, por otro, crear un metabuscador que emplee técnicas avanzadas de búsqueda como las mencionadas anteriormente (diversidad, relevance feedback y gestión de sesión).

1.2 Objetivos

Este trabajo apunta a dos objetivos básicos, generales: por un lado, el diseño y desarrollo de un metabuscador web con funcionalidades avanzadas, como diversidad y relevance feedback, que combine APIs de buscadores comerciales con el fin de explorar la aplicación de dichas técnicas sobre la búsqueda; por otro, desarrollar una plataforma que permita comparar y evaluar diversos buscadores, y llevar a cabo una comparativa de los mismos.

De forma concreta, los objetivos particulares abordados en el trabajo son los siguientes:

- Diseñar y desarrollar un metabuscador web, obteniendo resultados a partir de APIs de buscadores comerciales, como Google o Bing, combinándolos para proporcionar una respuesta unificada al usuario.
- Analizar las diferentes consultas de cada usuario con el fin de identificar sesiones (grupos de consultas relacionadas correspondientes a un mismo fin del usuario).
- Desarrollar mecanismos avanzados de búsqueda como relevance feedback o diversidad, orientados a mejorar determinados aspectos de los resultados obtenidos por el usuario al realizar la consulta y a mejorar la experiencia del usuario.
- Crear un sistema que permita combinar y evaluar diferentes sistemas de búsqueda. Sobre la base del metabuscador mencionado más arriba, diseñar e implementar una plataforma para evaluar y comparar distintos motores de búsqueda.
- Analizar los resultados de comparar entre sí diversos motores de búsqueda comerciales.

1.3 Estructura del trabajo

El presente documento se estructura de la siguiente forma:

- **Capítulo 1: Introducción.** En este capítulo se explican las motivaciones y objetivos del trabajo desarrollado.
- **Capítulo 2: Estado del arte:** En este capítulo se presentan los diferentes métodos y herramientas existentes para búsqueda web y evaluación de sistemas de recuperación de información.
- **Capítulo 3: Búsqueda y evaluación:** En este capítulo se detallan los métodos y algoritmos seleccionados para su implementados en la aplicación desarrollada en el presente trabajo.
- **Capítulo 4: Arquitectura y detalles de implementación:** En este capítulo se explica la arquitectura y el diseño de la aplicación, así como la interfaz desarrollada
- **Capítulo 5: Conclusiones.** En este capítulo, se presenta un resumen con las contribuciones del presente trabajo, así como las futuras líneas de trabajo.
- **Anexo 1: URLs y parámetros del servicio web:** Descripción de las diferentes llamadas al servicio web desarrollado, sus parámetros y sus retornos.

- **Anexo 2: Diagramas UML.** Diagramas de clases de los diferentes módulos de la aplicación desarrollada.

2. Estado del arte

La aparición de la Web ha motivado el desarrollo y la expansión de múltiples herramientas, tecnologías y algoritmos en el ámbito de la recuperación de información. En este capítulo, se proporciona una visión panorámica de algunas de estas técnicas, tanto en el campo de la búsqueda, como en el de la evaluación de sistemas de recuperación de información. En primer lugar, se estudian varias funcionalidades avanzadas de las que se puede aprovechar un buscador. En concreto, se explican metabúsqueda, gestión de sesión, relevance feedback y diversificación de resultados. Tras ello, se presentan métodos para realizar la evaluación simultánea de varios sistemas de recuperación de información.

2.1 Funcionalidades avanzadas de búsqueda

La búsqueda en entornos con cantidades masivas de información es uno de los principales problemas de la recuperación de información. Desde los años 90, con la aparición de la Web, teorías, herramientas y tecnologías relacionadas con los sistemas de búsqueda se han expandido enormemente. Buscadores como Google, Bing, Yandex o Baidu son algunas de las herramientas web más utilizadas hoy en día en todo el mundo. Este hecho propicia el desarrollo de nuevas técnicas para mejorar los resultados de los diferentes motores de búsqueda.

En esta sección se presentan y contextualizan algunas de estas técnicas y funcionalidades novedosas. En concreto, se estudian tres: La metabúsqueda, que permite combinar los resultados de diversos motores de búsqueda en un único ranking. Relevance feedback, que intenta mejorar los resultados obtenidos añadiendo palabras al final de una consulta. Por último, la diversificación de resultados, que reordena los resultados de forma que dos documentos similares no aparezcan juntos en el ranking.

2.1.1 Metabúsqueda

La información almacenada en la red está cambiando continuamente. Esto, unido a la escala masiva de la web, incomparable por varios órdenes de magnitud a cualquier repositorio de información anterior, causa que indexar todos los documentos la web sea un reto al alcance de pocos partícipes del mercado de las tecnologías de búsqueda y acceso a la información. Además, debido a la amplia variedad existente de modelos y algoritmos de búsqueda, los resultados recuperados por diversos motores de búsqueda pueden ser muy dispares y, en consecuencia, los documentos relevantes obtenidos por un buscador podrían no ser recuperados por otro. Spink et al. (2006) estimaron que, en media, cerca de un 70% de los documentos mostrados en la primera página de cuatro importantes buscadores web (Google, Yahoo, Ask y MSN Search) eran únicos de cada motor. Este hecho podría provocar que los usuarios tuviesen que realizar consultas en distintos motores de búsqueda para obtener aquella información que buscan. Para tratar de solucionar estas limitaciones, surgen técnicas como la metabúsqueda.

La metabúsqueda es una técnica que permite combinar rankings procedentes de diferentes motores de búsqueda para proporcionar a los usuarios una respuesta unificada. Dada una consulta determinada, un metabuscador lanza la consulta a un conjunto de buscadores. Cada uno de estos sistemas devuelve un ranking de resultados, que se combinan para proporcionar el resultado deseado.

La idea de combinar diferentes sistemas de búsqueda se introdujo a principios de los años 90, en el proyecto DARPA TIPSTER. Desde entonces, se han propuesto diferentes modos de llevar a cabo esta fusión, y hoy en día, buscadores web como DogPile, iZito, IxQuick, Search.com o eTools aplican esta técnica a partir de datos solicitados a diferentes motores de búsqueda comerciales, como el buscador de Google o Bing.

A diferencia de las herramientas básicas de búsqueda, mediante este método no es necesario disponer de un índice propio de la colección de documentos para obtener un ranking. Los documentos a devolver para una determinada consulta vienen dados por los rankings obtenidos por los diferentes motores de búsqueda agregados. Al realizar una fusión de dichos resultados, se evita que el usuario tenga que consultar los diferentes buscadores por separado para obtener los diferentes resultados, al presentarse todos los resultados en un ranking unificado que se obtiene desde un único punto de acceso.

Esta fusión de resultados, además, puede mejorar la precisión del buscador, al combinar todos aquellos documentos que cada motor, individualmente, considera buenos (Montague et al. 2001). Además, si el solapamiento entre los resultados procedentes de los distintos buscadores es bajo, la metabúsqueda puede abarcar un número mayor de documentos que cada uno de los motores agregados de forma individual.

La metabúsqueda plantea dos problemas a resolver. Por un lado, la selección de fuentes, y por otro, la fusión de rankings.

2.1.1.1 Selección de buscadores

La principal razón para emplear la metabúsqueda radica en las diferencias existentes entre los distintos motores de búsqueda. Dados dos sistemas distintos, conviene fusionarlos en caso de que cada uno de ellos proporcione diferentes resultados. Si los resultados fueran muy similares, la ganancia de información al combinar dos resultados es obviamente prácticamente nula. Esto es debido a que la combinación de dos rankings idénticos no presenta diferencias con cada uno de los rankings individuales. La combinación es interesante con sistemas que, o bien tengan un solapamiento bajo de documentos indexados, o bien devuelvan resultados dispares en determinadas consultas concretas.

2.1.1.2 Fusión de rankings

La fusión o agregación de rankings consiste en la combinación de diversos rankings, previamente obtenidos o calculados, en una única lista que contiene, sin repeticiones, todos los elementos de los rankings iniciales. Se trata de una operación omnipresente en diversos campos de la recuperación de información, no solo en metabúsqueda, sino también en búsqueda personalizada, consenso de preferencias de distintos usuarios, clasificación mediante clasificadores *ensemble* (uniones de clasificadores), o en sistemas de recomendación. Es muy común en sistemas de búsqueda orientados a la web emplear esta técnica, ya sea para combinar diferentes señales procedentes de múltiples algoritmos y modelos (por ejemplo, Google combina más de 200 señales diferentes antes de mostrar sus resultados), o para metabúsqueda.

Existen métodos muy variados para realizar esta fusión, pero, en función de la información empleada para combinar los resultados, podemos clasificarlos en dos tipos (Lee 1997):

- **Métodos basados en score:** Estos métodos parten de los *scores* obtenidos por los distintos motores de búsqueda para cada resultado. Dado que estas puntuaciones pueden ser muy variadas, es necesario normalizar los *scores* obtenidos por cada buscador previamente a la combinación (Montague et al. 2001). Para ello, se pueden emplear diferentes métodos, como *min-max* o *z-score* (Markov et al. 2012).
- **Métodos basados en posición:** Estos métodos parten de las posiciones de cada uno de los documentos en los rankings de cada uno de los buscadores. A diferencia de los anteriores, estos métodos pueden aplicarse siempre para fusionar diversos rankings, ya que, aparte de conocer los documentos que conforman los mismos y su orden, no se necesita información adicional. Para realizar esta fusión, existen métodos como Borda o Rank-Sim (Lee 1997), que permiten obtener *scores* para cada resultado, o métodos que emplean cadenas de Markov para combinar directamente los resultados.

Una vez obtenidos los *scores* normalizados, tanto para el primer tipo como para el segundo, es necesario combinar los resultados de alguna manera. Los métodos de combinación posibles

son muy variados, y comprenden desde una combinación lineal ponderada de los *scores*, hasta otros más complejos como regresión lineal, o métodos basados en cadenas de Markov.

2.1.2 Gestión de sesión y relevance feedback

La diversidad de usuarios de un buscador web provoca que, ante una misma consulta, las necesidades de información de cada uno de los usuarios sean muy diferentes. Por ello, actualmente, buscadores como Google o Faroo emplean técnicas para personalizar los resultados ofrecidos. Para ello, se tienen en cuenta diversos factores y observaciones de la actividad del usuario por parte del sistema, tales como consultas anteriores, documentos seleccionados por los usuarios, etc.

En este marco de personalización de los resultados, se han concebido técnicas como relevance feedback, que permiten emplear el comportamiento previo del usuario para refinar internamente sus consultas, tratando de ajustarse mejor a las necesidades individuales del usuario implícitamente inferibles de su comportamiento. El estudio de este comportamiento previo puede ser estudiado de diversas formas. En este proyecto, se emplearán las llamadas sesiones de búsqueda.

En este apartado, se presentan tanto las funcionalidades de análisis y gestión de las sesiones de búsqueda desarrolladas en el presente TFG, como las técnicas de relevance feedback implementadas en el mismo, y se explicarán las soluciones adoptadas.

2.1.2.1 Sesiones de búsqueda

Las técnicas tradicionales de búsqueda Web tratan cada consulta como una operación aislada. Sin embargo, otra perspectiva consiste en tratar una búsqueda como una secuencia de consultas, a través de las cuales el usuario va, por ejemplo, acotando lo que buscaba, reorientando sus objetivos, o encontrando diferentes piezas de una búsqueda que tiene varias partes. Es por ello que surge el concepto de sesión de búsqueda.

Si bien en la literatura se han propuesto diferentes definiciones para este concepto (Gayo-Avello 2009), todas ellas se pueden resumir en la siguiente: una sesión de búsqueda es una secuencia de consultas orientadas a satisfacer una única necesidad de información (Jansen et al. 2007). El análisis de estas sesiones de búsqueda permite modelizar el comportamiento de los diferentes usuarios de un determinado motor de búsqueda, y por ende la identificación y gestión de las mismas tiene interés para añadir funcionalidades a estos sistemas, como la personalización de los resultados.

La detección de las fronteras entre las distintas sesiones de búsqueda es un problema abierto en absoluto trivial, ampliamente estudiado en el área. Con el fin de realizar la clasificación de una consulta entre varias sesiones, se han desarrollado multiples métodos. A continuación, se detallan algunos de ellos:

Métodos temporales

Estos métodos se basan en la proximidad temporal de las diferentes consultas en una misma sesión. Debido a su sencillez, es un método muy usado para diferenciar entre sesiones.

Para clasificar una consulta en una sesión determinada, se establece un cierto límite en el lapso temporal entre consultas, que suele ir entre 5 y 30 minutos, aunque en algunos casos, se ha llegado a contemplar hasta 120 minutos (Hagen et al. 2011). Si la diferencia temporal entre una consulta realizada por el usuario y la anterior es menor que dicho límite, se considerará que ambas consultas forman parte de una misma sesión. En caso contrario, se considerará que la nueva consulta está dentro de una nueva sesión.

Métodos basados en la consulta (reconocimiento de patrones)

Estos métodos utilizan la información contenida en el texto de la consulta para diferenciar entre dos posibles sesiones. Dadas dos consultas consecutivas, q_i y q_{i+1}, se comparan, y se clasifican en una de las siguientes categorías.

- **Repetición:** Ambas consultas son iguales.
  Ejemplo: q_i = UAM, q_{i+1} = UAM.
- **Especialización:** q_{i+1} busca información más específica que q_i (Se han añadido términos a la consulta q_i).
  Ejemplo: q_i = Ingeniería, q_{i+1} =Ingeniería informática.
- **Generalización:** q_{i+1} busca información más general que q_i (Se han eliminado términos de la consulta q_{i+1}).
  Ejemplo: q_i = Ingeniería informática, q_{i+1} =Ingeniería
- **Reformulación:** Se han añadido algunos términos a q_{i+1}, y se han eliminado otros de q_i, pero tienen términos en común.
  Ejemplo: q_i = Ingeniería informática, q_{i+1} =Ingeniería de telecomunicación.
- **Nuevo:** No hay elementos comunes explícitos entre las consultas.
  Ejemplo: q_i = Escuela Politécnica, q_{i+1} = Universidad Autónoma.

Otras clasificaciones añaden más categorías, en función de las distintas acciones que puede llevar a cabo el buscador, como la paginación (He et al. 2002).

Habitualmente, si este método se utiliza en solitario, se especifica que ambas consultas pertenecen a distintas sesiones si la comparación se clasifica como *Nuevo*, y, en caso contrario, se establece que ambas consultas pertenecen a la misma sesión (Jansen et al. 2007).

En el caso de que se obtenga cualquiera de las cuatro primeras categorías, parece casi seguro que las consultas pertenecen a la misma sesión. Sin embargo, aparece un problema a la hora de realizar la clasificación. Por ejemplo, si se analizan las consultas "Java Programming Language" y "Java Coffee", la necesidad de información es distinta, y ambas consultas deberían pertenecer a sesiones distintas, pero, sin embargo, se añade la consulta "Java Coffee" a la sesión iniciada, puesto que la comparación se clasificaría como *Reformulación.* Por otro lado, si la comparación se clasifica como *Nuevo*, es posible que la segunda consulta pertenezca semánticamente a la misma sesión (ejemplo: Universidad Autónoma de Madrid, UAM). Debido a ello, en caso de que la comparación se clasifique en *Nuevo* o *Reformulación*, se suelen aplicar otras estrategias (Hagen et al. 2011).

Métodos de aprendizaje automático

Estos métodos parten de los resultados anteriores (el tiempo entre sesiones y la clasificación de consultas mediante patrones), combinándolos mediante técnicas de aprendizaje automático. A partir de un conjunto de datos clasificado por expertos, se entrena un determinado modelo, que, a partir de dicha clasificación y la diferencia temporal entre sesiones, trata de identificar si hay o no un cambio de sesión. Se han utilizado diferentes modelos para ello, como redes neuronales, regresión lineal múltiple o simulación de Montecarlo.

Métodos geométricos

Gayo-Avello (2009) propone un método distinto, combinando de nuevo el tiempo entre consultas y el reconocimiento de patrones. Para ello, se establece:

- Un tiempo máximo. Por encima de este tiempo t, las sesiones serán distintas
- Una distancia léxica entre las cadenas de dos consultas, que valga 0 para consultas idénticas, y 1 para consultas completamente distintas.

Llamando A al caso en el que la distancia léxica entre las consultas es máxima, pero no hay diferencia de tiempo entre ellas (punto $(0,1)$), y B al caso de consultas totalmente idénticas realizadas con distancia temporal t (punto $(t, 0)$), definimos una curva que pasa por los puntos A y B.

Si el par (distancia léxica, tiempo) se encuentra bajo la gráfica, consideraremos que ambas consultas pertenecen a la misma sesión, y si no, consideraremos que pertenecen a sesiones disjuntas. Esto se ilustra en la Figura 1.

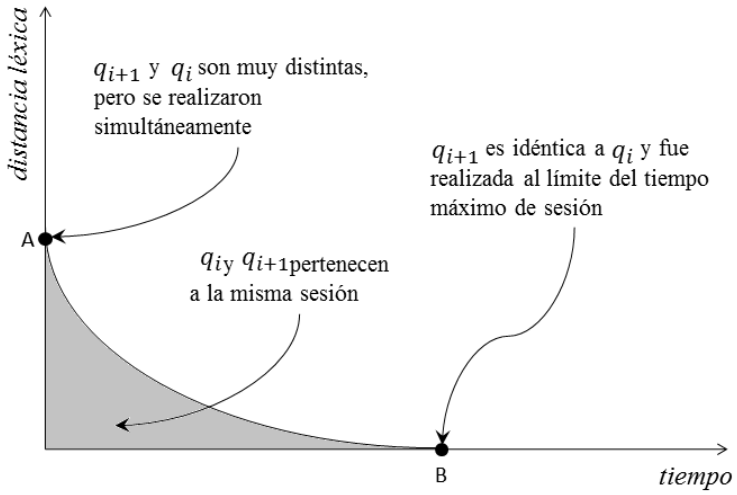

Figura 1. Método geométrico para detección de sesión

Métodos semánticos

Otro método para realizar la detección de sesión se basa en el llamado análisis semántico explícito (*Explicit Semantic Analisis*, ESA, Hagen et al. 2011). Este método parte de un conjunto de documentos previamente almacenados (por ejemplo, de Wikipedia). Estos documentos se van a tomar como vectores de un espacio vectorial, con coordenadas dependientes diversos factores, como la frecuencia de aparición o el número de documentos en los que aparece. Con dichos vectores se crea una matriz. A esa matriz se la llamará matriz ESA.

Tras ello, se halla, por un lado, un vector formado por las palabras clave de la nueva, y, por otro, un vector formado por las palabras clave de las consultas de la sesión en la que se encuentra la consulta anterior de ese usuario. Ambos vectores se multiplican por la matriz ESA, y se calcula la similitud del coseno entre los vectores resultantes. Si la similitud es mayor que un determinado umbral, se considera que ambas consultas pertenecen a la misma sesión. Si no, se considera que pertenecen a diferentes sesiones de búsqueda.

Otro método semántico para detectar sesiones emplea también la similitud del coseno. Este método crea, para la nueva consulta q_{i+1} como para la anterior, un vector formado a partir de las palabras contenidas en los títulos y *snippets* de los documentos recuperados por el buscador. Una vez hecho esto, se calcula la similitud entre ambos vectores mediante el coseno. Si este coseno supera un umbral establecido, se considera que ambas consultas pertenecen a la misma sesión. En caso contrario, se considera que la consulta q_{i+1} pertenece a una nueva sesión.

Otros métodos

Además de los métodos mencionados anteriormente, existen otros. Por ejemplo, aparece la posibilidad de combinar diferentes métodos de detección de los cambios de sesión para mejorar el acierto. Hagen et al. (2011) proponen, por ejemplo, un método que combina, en cascada, reconocimiento de patrones, un método geométrico, análisis semántico de la consulta, y, por último, estudia la existencia de URLs en común.

2.1.2.2 Relevance Feedback

Los usuarios de los diferentes motores de búsqueda web a menudo encuentran dificultades a la hora de expresar sus necesidades de información en forma de una consulta. Con el fin de ayudar a estos usuarios a encontrar aquello que buscan, surge relevance feedback. Relevance feedback cubre una serie de técnicas cuyo objetivo es mejorar la consulta de un usuario, usando más información (*feedback*) del usuario que la consulta explícita, y facilitar de ese modo la recuperación de información (Ruthven et al. 2003).

Dada una consulta inicial de un usuario, las diferentes técnicas existentes de relevance feedback extienden la consulta, añadiendo palabras a partir de las acciones llevadas a cabo por el usuario anteriormente, o modificando los pesos de los términos de la consulta. Si bien los comienzos de estas técnicas datan de los años 70, su uso en buscadores web no está muy extendido, y la aplicación de algunas de estas técnicas en sistemas de búsqueda sigue siendo actualmente un área abierta, lo que motiva el interés por explorarla en el presente TFG. El procedimiento básico de relevance feedback es el siguiente:

1. El usuario ejecuta una consulta inicial.
2. Los resultados para la misma son presentados al usuario, que los explora, e indica aquellos resultados que considera relevantes.
3. El sistema genera puntuaciones para los términos que aparecen en los diferentes documentos relevantes, generando un ranking de palabras.
4. Los m primeros resultados de dicha lista se añaden a la consulta, para posteriormente, ajustar los pesos de cada uno de los términos de la nueva consulta, y solicitar al sistema de búsqueda los resultados correspondientes a la nueva petición.
5. Se presentan los resultados al usuario. A este proceso se conoce como iteración de relevance feedback.

Un ejemplo de estas iteraciones podría ser el siguiente: un usuario busca "Escuela Politécnica" en un buscador. Tras seleccionar los resultados relevantes para su búsqueda, el sistema calcula aquellos términos que considera más relevantes dentro del texto que contienen los documentos, y los añade a la consulta. Por ejemplo, añadiendo dos palabras, la nueva consulta podría ser la siguiente: "Escuela Politécnica Superior UAM".

Existen diversos métodos para añadir términos a una consulta. Generalmente, estos métodos dependen del modelo de recuperación de información empleado por los buscadores. A continuación, se estudiarán los métodos existentes de relevance feedback para dos de ellos, el modelo booleano y el vectorial:

Modelo booleano

En este modelo, las palabras se ven como conjuntos de documentos (aquellos documentos que contienen al término). Las consultas son operaciones lógicas booleanas entre dichos conjuntos, y se devuelven aquellos documentos que cumplen la condición expresada. Por ejemplo, en la consulta "Trabajo OR (Estudios AND Beca)", se recuperarían todos aquellos documentos que contuviesen o bien la palabra "Trabajo" o bien las palabras "Estudios" y "Beca".

Si bien en el modelo booleano no se consideran las frecuencias de los términos, un posible método para aplicar relevance feedback en este modelo sería recuperar aquellos términos que aparecen más veces en documentos relevantes que en documentos no relevantes. Una vez hecho esto, se podrían presentar los resultados al usuario para que seleccionase las que más se aproximan a su búsqueda, o añadir las palabras directamente a la consulta (Ruthven et al. 2003).

Modelo vectorial

En este modelo, las palabras se consideran ejes de un espacio vectorial. En este espacio, los documentos se representan mediante vectores, con coordenadas dependientes de diversos factores, como la frecuencia de aparición de un término en el documento, o el número de documentos en los que aparece. En este caso, se emplea como función de ranking una función similitud por coseno al vector de la consulta. Es decir, se calcula un *score* de la siguiente manera:

$$score(d,q) = cos(\vec{d},\vec{q}) = \frac{\vec{d}\cdot\vec{q}}{|\vec{d}||\vec{q}|} = \frac{\sum_t d_t q_t}{\sqrt{\sum_t d_t^2}\sqrt{\sum_t q_t^2}}$$

donde $t \in \{1,2,\dots,|\mathcal{V}|\}$, y $|\mathcal{V}|$ es el tamaño del vocabulario (y, por tanto, la dimensión del espacio vectorial). Tras ello, se ordenan todos los documentos de mayor a menor score para generar un ranking de documentos.

Es en este modelo de recuperación de información donde se propone en los años 70 la primera técnica de relevance feedback, por parte de J.J. Rocchio (1971). Esta técnica formula la nueva consulta de la siguiente manera:

$$\overrightarrow{Q_1} = \overrightarrow{Q_0} + \frac{1}{|Rel|}\sum_{d \in Rel} \vec{d} - \frac{1}{|\neg Rel|}\sum_{d \in \neg Rel} \vec{d}$$

donde Rel es el conjunto de documentos relevantes, y $\neg Rel$ es el conjunto de documentos no relevantes.

Este nuevo vector no solamente añade términos a la consulta original, sino que además, modifica los pesos de los términos de dicha consulta, provocando que aquellos términos que aparezcan más veces en documentos relevantes que en documentos no relevantes tengan mayor importancia. En caso de que algún término tuviese coordenada negativa o 0 en la nueva consulta, se eliminaría de la misma.

Otras técnicas de relevance feedback aplicadas sobre el modelo vectorial surgen a partir de modificaciones sobre la fórmula anterior, tomando diferentes pesos para los documentos relevantes y los no relevantes, o restando únicamente el primer documento no relevante obtenido en el ranking.

Todos los métodos mencionados anteriormente tienen una característica en común: emplean aquellos documentos que el usuario ha indicado previamente como relevantes. Sin embargo, existe la posibilidad de aplicar técnicas para extender la consulta sin necesidad de que el cliente haya proporcionado tal información. Esta variante se conoce como *pseudo-relevance feedback*, o *blind relevance feedback* (Ruthven et al. 2003).

Esta modalidad considera que los primeros k documentos devueltos por un buscador a partir de la consulta inicial son relevantes, y, a partir de ello, se aplica una iteración de relevance feedback para ampliar la consulta. La idea de la que parten estas técnicas es la siguiente: A partir de los documentos más similares a la consulta inicial, aplicar relevance feedback podría proporcionar un mejor ranking.

Esta aproximación puede producir un problema a la hora de extender la consulta, conocido como *query drift*: En caso de que la lista de resultados no contenga documentos relevantes, o contenga pocos, los términos que se añadirán posteriormente a la consulta no identificarán correctamente los documentos relevantes. Es decir, en caso de que la primera respuesta a la consulta inicial fuese "mala", es decir, el buscador no devolviese documentos relevantes, los métodos de pseudo-relevance feedback funcionarían mal.

2.1.3 Diversidad

Las consultas realizadas sobre un buscador web a menudo son ambiguas. Por ejemplo, la consulta 'Java' puede estar referida al lenguaje de programación, a la isla de Indonesia, o incluso a las vainas de café procedentes de dicha isla. En la respuesta de un buscador a una consulta, los documentos seleccionados en el ranking tieneden a ser muy similares entre sí, y no es difícil que únicamente se muestren resultados relativos a una de estas interpretaciones posibles. Como solución para paliar estos problemas, se ha planteado hace algo más de una década la perspectiva de diversificar la lista de documentos devueltos.

A diferencia de las técnicas clásicas de búsqueda, que tratan de maximizar el número de documentos relevantes dada una consulta, la diversificación de resultados tiene otro objetivo: maximizar la probabilidad de que el usuario obtenga al menos un documento relevante tras ejecutar la consulta (Chapelle et al. 2011). Aumentando la diversidad de los resultados mostrados y penalizando resultados muy similares en posiciones demasiado cercanas del ranking, es más probable que alguno de ellos interese al usuario sin tener que descender más de lo deseable en la lista de resultados.

La aproximación típica para diversificar resultados consiste en partir de un ranking inicial devuelto por un algoritmo de búsqueda estándar ante una consulta. A partir de los resultados

iniciales, se reordenan los documentos generando un nuevo orden que disminuya la similitud entre los documentos próximos. Existen dos perspectivas diferentes para realizar esta diversificación: diversificación basada en similitud, y diversificación basada en aspectos.

2.1.3.1 Diversificación basada en similitud

Esta perspectiva asume que dos documentos cubren temas o aspectos similares si son similares entre sí. Esta es la perspectiva seguida por diversos autores, como Carbonell (1998). Aplicar estos métodos para realizar la diversificación requiere abordar dos problemas: En primer lugar, se debe escoger una función de similitud entre documentos. El segundo problema es la diversificación del ranking.

Función de similitud

Las técnicas de diversidad penalizan aquellos documentos muy similares a los mostrados anteriormente en el ranking. Ante ello, es necesario establecer qué significa que dos resultados sean similares. Para ello, se define una función de similitud entre dos documentos.

Existen múltiples funciones que se pueden emplear a la hora de comprobar si dos documentos son similares. Por ejemplo, la similitud del coseno empleada para recuperar documentos en el modelo vectorial se puede emplear para comparar documentos (Carbonell et al. 1998), así como otras medidas, como la similitud de Jaccard o la similitud de Dice.

Diversificación del ranking

El objetivo de las técnicas de diversificación es obtener un nuevo ranking cuyos documentos sean más diversos que en el *ranking* inicial proporcionado por el buscador. No obstante, no se debe dar de lado esta lista de partida por completo, pues podría ser que algunos de estos documentos relevantes se recuperasen más tarde.

Carbonell et al. (1998) propusieron un método de diversificación llamado *Maximal Marginal Relevance* (MMR), que trata de realizar una suma ponderada entre la puntuación obtenida por el documento en el ranking, y la distancia a los documentos del ranking. El método es el siguiente:

1. Se calculan las similitudes entre cada uno de los pares de documentos del *ranking* inicial.
2. Se selecciona el primer elemento del antiguo ranking, y se coloca como primer elemento del nuevo.
3. Para cada documento no añadido al ranking anteriormente, se calcula la relevancia marginal (MR), que se define como:

$$MR = (1-\lambda) f(d_i, q) - \lambda \max_{d_j \in S} sim(d_i, d_j)$$

   donde S son los documentos añadidos al nuevo ranking, y $f(d_i, q)$ es el score del documento i-ésimo en el ranking inicial, R para la consulta q. λ es un parámetro que toma valores entre 0 y 1, y $sim(d_i, d_j)$ es la función de similitud entre documentos.
4. Se escoge el elemento que maximice la relevancia marginal, y se añade al ranking.
5. Se itera desde el paso 3 hasta que no queden más elementos por añadir.

Vista la fórmula anterior, se observa que el primer término de la resta tiene como objetivo aprovechar el *score* del documento en el ranking antiguo para construir el nuevo, mientras que el segundo término mide la disparidad entre el documento que se quiere añadir al ranking y los documentos ya añadidos. Este método obtiene, para $\lambda = 0$, el ranking original, mientras que, para $\lambda = 1$, se obtiene un ranking completamente diversificado, pero que no tiene en cuenta los scores iniciales de cada documento.

A la hora de escoger el parámetro λ, existen diversas opciones. Una primera opción consiste en utilizar un valor fijo entre 0 y 1 para todas las consultas. Sin embargo, dado que no todas las consultas son igual de ambiguas (por ejemplo, 'UAM' es una consulta más ambigua que 'Universidad Autónoma de Madrid', dado que estas siglas se pueden emplear para otras entidades, como la Universidad Autónoma Metropolitana), puede ser interesante variar el valor de este

parámetro en función de la consulta realizada. Santos et al. (2010 A) proponen un método basado en vecinos próximos para calcular este parámetro en cada consulta: Dada una consulta, se obtienen aquellas k consultas más similares a esta, y se calcula como parámetro λ la media de los parámetros para estas consultas similares:

$$\lambda = \frac{1}{k} \sum_{i|\ q_i \in N_k(q)} \lambda_i$$

donde $N_k(q)$ son las k consultas del conjunto de entrenamiento más similares a la consulta q, de acuerdo a una determinada métrica, como la distancia euclídea.

2.1.3.2 Diversificación basada en aspectos

La diversificación basada en aspectos trata de modelar las distintas interpretaciones posibles para una consulta, buscando proporcionar, mediante los documentos, la mayor cobertura de las mismas (Santos et al, 2010 B). Existen diferentes algoritmos para realizar esta diversificación, como xQuAD o IA-SELECT, que se explican a continuación:

xQuAD (Explicit Query Aspect Diversification)

Santos et al, (2010 B) desarrollaron este método para llevar a cabo la diversificación de resultados. Para identificar los diferentes aspectos de la consulta, se emplean las consultas derivadas o sugeridas que proporcionan hoy en día múltiples buscadores, como Google o DuckDuckGo.

Para crear el ranking diversificado, se escoge el documento no seleccionado previamente que maximice la siguiente fórmula:

$$(1-\lambda)P(d|q) + \lambda P(d, \bar{S}|q)$$

En la fórmula anterior, $P(d|q)$ es la probabilidad de que aparezca el documento dado la consulta inicial, y modela la relevancia del documento. $P(d, \bar{S}|q)$ es la probabilidad de que el documento aparezca y que los documentos previamente introducidos en el ranking diversificado, S, no aparezcan, dada la consulta inicial, y modela la diversidad de los documentos.

Para aprovechar las consultas derivadas, la segunda probabilidad en la fórmula anterior se calcula de la siguiente manera:

$$P(d, S|q) = \sum_{q_i \in Q} P(q_i|q) P(d, \bar{S}|q_i)$$

donde $P(q_i|q)$ es una medida de la importancia de la subconsulta q_i para la consulta q, Q es el conjunto de las subconsultas obtenidas, y se cumple que:

$$\sum_{q_i \in Q} P(q_i|q) = 1$$

y

$$P(d, \bar{S}|q_i) = P(d|q_i) P(\bar{S}|q_i)$$

IA-Select

Agrawal et al. (2009) propusieron este método, que parte de una clasificación de los documentos devueltos en una consulta en diferentes categorías. Estas categorías no realizan una partición de los documentos, sino que un mismo documento puede pertenecer a varias categorías para una misma consulta. El algoritmo es el siguiente:

1. Se calcula, para cada documento, $V(d|c, q)$, que es la probabilidad de que el documento satisfaga la necesidad del usuario dada la consulta y la suposición de que el usuario busca documentos de la categoría c.

2. Se calcula $U(c|q,S) = P(c|q)$ para cada categoría. $U(c|q,S)$ va a representar la probabilidad de escoger la categoría c, dado que los documentos previamente añadidos al ranking diversificado no satisfacen la necesidad del cliente.
3. Mientras queden documentos a agregar al ranking diversificado:
   a. Se calcula, para cada documento y categoría la función
   $$g(d|q,c,S) = U(c|q,S)P(c|q)$$
   donde S es el ranking diversificado.
   b. Se escoge el documento que obtenga el máximo de $g(d|q,c,S)$, y se añade al ranking S
   c. Se actualiza $U(c|S)$ mediante la regla de Bayes.

Este método es óptimo si cada documento pertenece a una única categoría.

2.2 Evaluación

La evaluación de un motor de búsqueda es una perspectiva particular del problema general de valorar y medir la efectividad de un sistema de recuperación de información. La evaluación ha sido un área de atención prioritaria en el campo de recuperación de información, en la que se ha desarrollado un corpus muy importante de metodologías, métricas, paradigmas y fundamentos estrechamente ligados al desarrollo de algoritmos y soluciones de búsqueda, orientando la definición e investigación de estos, y el desarrollo del campo en general. El objetivo último de un método de evaluación es medir cómo de bueno es un sistema frente a otro, estudiando comparativamente características de los diferentes sistemas como el número de documentos relevantes devueltos, las posiciones en las que los buscadores devuelven dichos documentos, etc. Naturalmente no existe una dimensión ni una definición única de lo que es un sistema efectivo, pues por una parte depende de la finalidad y uso específico que se vaya a hacer de él, y por otro las bondades de un sistema consisten en una suma de cualidades, algunas de las cuales pueden incluso ser mutuamente contrapuestas, en cuyo caso se trataría de obtener un compromiso óptimo o satisfactorio.

Tradicionalmente, las técnicas de evaluación han girado en torno a la noción de relevancia, que suele definirse como la propiedad que cumple un documento que satisface una necesidad de información para la que se ha formulado una consulta. A nivel práctico en la labor experimental, esta noción se materializa mediante los llamados juicios de relevancia, que indican, para unas determinadas consultas, se indica qué documentos son aquellos que deberían ser devueltos por el buscador. Un ejemplo de ello son las colecciones proporcionadas por la Text REtrieval Conference (TREC). Estas colecciones constan de un conjunto de documentos, junto a una serie de consultas, cada una de las cuales tiene asociado un conjunto de documentos relevantes. A partir de ellos, es posible realizar una evaluación sobre lo bien que funciona un algoritmo o técnica de búsqueda mediante el cálculo de diferentes métricas, que no son otra cosa que funciones sobre una lista de resultados con los correspondientes juicios de relevancia.

Esta metodología de evaluación se sigue realizando por empresas importantes en el sector de la búsqueda web, como Google, pero solamente supone el comienzo de la evaluación de los sistemas. Esto se debe a que los juicios de relevancia obtenidos podrían depender de diversos factores, como el usuario del sistema, el marco espacio-temporal en el que se realizan las consultas, etc.

Una idea más novedosa para realizar estas evaluaciones consiste en recoger *feedback* que los usuarios del sistema generan en el uso natural del sistema en producción, obteniendo información de los *clicks* realizados por los mismos. En este apartado, se exploran distintas vías de comparación de diversos buscadores mediante los *clicks* de los usuarios. En concreto, se estudian los tests A/B.

Los tests A/B permiten comparar dos sistemas, A y B, a partir de la interacción (en este caso clicks) realizada por los usuarios sobre las salidas (rankings de resultados) de los sistemas que se someten a evaluación. Se trata de una técnica omnipresente en la evaluación de sistemas de

recuperación de información, tanto en búsqueda como en otros campos, como la recomendación. Habitualmente, se emplea para comparar variantes de un sistema, siendo A el sistema antiguo, y B el nuevo sistema que se quiere probar para decidir si se despliega o no.

Este tipo de tests no presupone un conocimiento completo o exacto de la relevancia de los documentos, sino que trata los documentos seleccionados por el usuario como aquellos posiblemente relevantes. Bajo este supuesto, el principio básico para decidir qué sistema es mejor se basa en términos generales en observar qué version, A o B, recibe más clicks.

La forma en la que se articulan los test A/B de los buscadores comerciales adoptan una de las dos opciones siguientes: a) redirigir parte del tráfico de (es decir las consultas recibidas por) el buscador al sistema B, y obtener por separado las valoraciones de los dos sistemas; o bien b) dirigir todas las consultas del experimento a ambos sistemas, e intercalar los resultados en un único ranking, y atribuir los clicks a un sistema u otro en función de cuál de los dos devolvió el resultado clickado, con diversas estrategias para dilucidar la intersección entre los dos buscadores. Este segundo método es más avanzado que el primero y ha empezado a investigarse y emplearse recientemente. Es por ello que se ha considerado de interés y objeto de estudio e implementación en el presente trabajo.

Si bien es cierto que el intercalado de resultados busca generar un ranking unificado de documentos, los métodos para hacerlo difieren de los métodos de combinación para la metabúsqueda mostrados en la sección 2.1.1. En este caso, no se trata de ordenar los resultados siguiendo un criterio de relevancia, sino que se trata de ordenar los resultados de forma que no se otorgue ventaja por posición a ninguno de los buscadores a evaluar, para no sesgar los resultados del test. Decimos que un método está sesgado si, bajo una sucesión aleatoria de clicks, es más probable que a un ranking se le asignen más clicks que al otro. Si un método de evaluación está sesgado, el ruido existente en los clicks estudiados como feedback de los usuarios puede afectar a la salida esperada de la comparación, lo que produce que el método de evaluación sea poco fiable (Hoffman et al., 2011).

A la hora de realizar el intercalado de resultados, existen dos problemas a tratar. El primer problema es la forma de combinar los distintos rankings para generar la lista de resultados a mostrar. El segundo consiste en establecer el modo de atribuir los clicks realizados sobre la lista unificada a cada sistema. Estos dos problemas no se resuelven por separado, sino que cada método de intercalado de resultados está a menudo relacionado con un método de asignación de clicks. A continuación se presentan algunas de las técnicas de intercalado de dos rankings más comunes para tests A/B.

2.2.1 Método balanceado

Este método toma como entrada dos listas de resultados, l_1 y l_2, y las combina de la siguiente manera: En primer lugar, se escoge de aleatoriamente una de las dos listas, y se toma el primer documento de la misma. Este documento se sitúa como el primero del nuevo ranking. Una vez hecho esto, se escoge el primer documento del segundo ranking. Si no pertenece a la lista intercalada, se añade. Una vez hecho esto, se observa el segundo elemento de la primera lista, y se procede igual que antes hasta que todos los documentos pertenezcan al ranking final.

En cuanto a la asignación de clicks, para cada lista se halla la posición del elemento que ha recibido un click que se encuentre peor posicionado en el ranking. Una vez hecho esto, se halla el mínimo de las posiciones, k. Para cada ranking, por cada documento seleccionado por encima de k, se le atribuye un click. La lista ganadora es aquella con más documentos seleccionados en las posiciones del ranking situadas entre la primera y la k-ésima posición. Se muestra un ejemplo en la Figura 2.

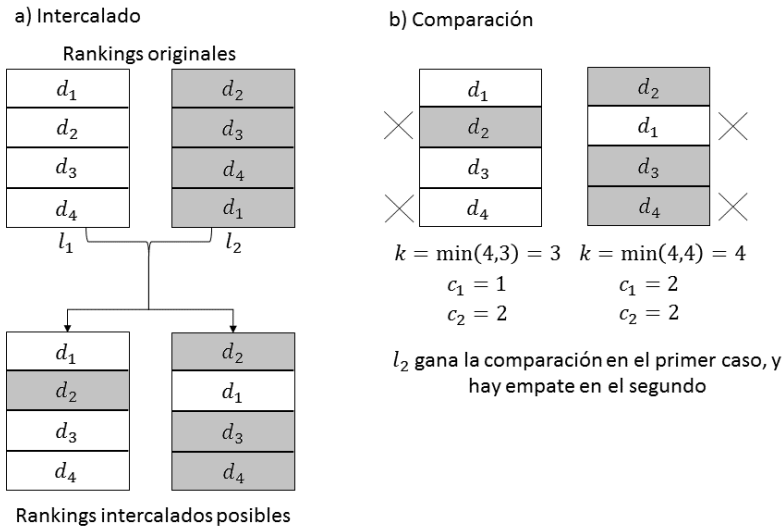

Figura 2. Método balanceado para el intercalado de rankings

Este método produce un sesgo para casos en los que las listas son muy similares (Hoffman et al. 2011). Por ejemplo, en el caso mostrado en la Figura 2, la primera lista obtenida es exactamente igual a la lista l_1, lo que provoca que los clicks realizados sobre los primeros documentos beneficien a la misma. Dado que los primeros documentos tienen mayor tendencia a ser seleccionados por los usuarios, esto produce que haya un sesgo hacia l_1.

2.2.2 Método team draft

Este método toma dos rankings l_1 y l_2, y los combina de la siguiente manera: para cada par de documentos a añadir a la lista, se escoge al azar una de las dos listas originales. De esta lista, se escoge el primer documento que no haya sido previamente añadido al ranking. Tras ello, se hace lo mismo con la segunda lista. Se repite el proceso hasta que no queden elementos por añadir de ninguno de los rankings iniciales.

Los clicks que se realicen sobre la lista unificada se atribuyen a la variante del buscador del que ha sido obtenido el documento seleccionado. Para cada ranking, se contará el número de clicks, y el ganador será aquel que más clicks haya recibido. Se muestra un ejemplo en la Figura 3.

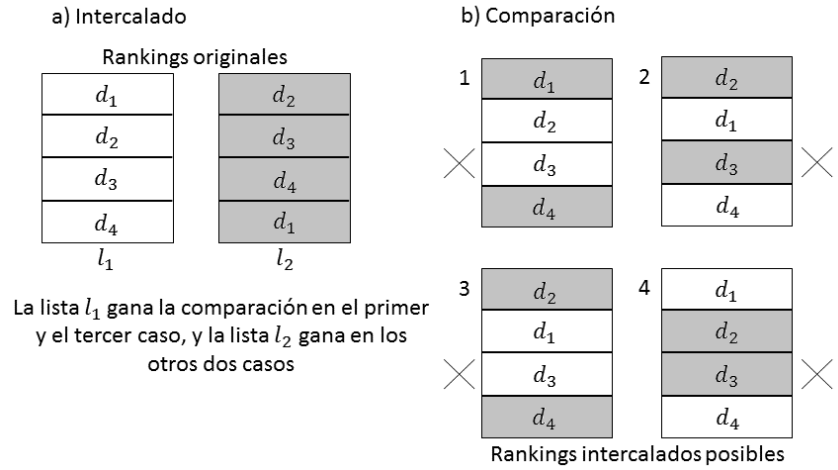

Figura 3. Método Team draft para el intercalado de rankings

Dado que a cada documento se le asigna un único ranking de partida, se soluciona el problema del sesgo que aparece en el método balanceado (Hoffman et al. 2011).

2.2.3 Método probabilístico

Hoffman et al. (2011) propusieron un método más sofisticado de intercalado. El intercalado se realiza de la siguiente forma: Previamente a la elección de un documento, se construye una función softmax para cada uno de los rankings. Esta función softmax asigna a cada documento del ranking una probabilidad de ser escogido. Una vez generadas las funciones, se escoge uno de los sistemas al azar. De ese sistema, se escoge un documento con probabilidad igual a la proporcionada por la función softmax. Una vez escogido un documento, se incorpora al ranking combinado, se elimina de las listas originales, y se normalizan las funciones softmax para cada buscador teniendo en cuenta los documentos de los rankings originales que restan por seleccionar. Este proceso se realiza hasta que se complete la lista unificada. En la Figura 4 se muestra un ejemplo de esta forma de intercalar documentos.

Mediante este método, si un documento aparece entre los primeros resultados de los dos rankings a intercalar, la probabilidad de que el documento aparezca en una posición más alta en el ranking combinado aumenta, algo que no ocurre en otros métodos, como team draft.

A la hora de atribuir los clicks a cada uno de los documentos, existen diversas posibilidades. En primer lugar, es posible realizar una atribución de clicks similar a la de team draft, es decir, atribuir cada click al ranking del que procede el documento seleccionado. Una segunda posibilidad consiste en estimar el número de clicks esperado para el ranking (Hoffman et al. 2011). Aplicando cualquiera de estos dos sistemas, es posible demostrar que ninguno de ellos presenta un sesgo a la hora de evaluar los diferentes buscadores (Hoffman et al. 2013).

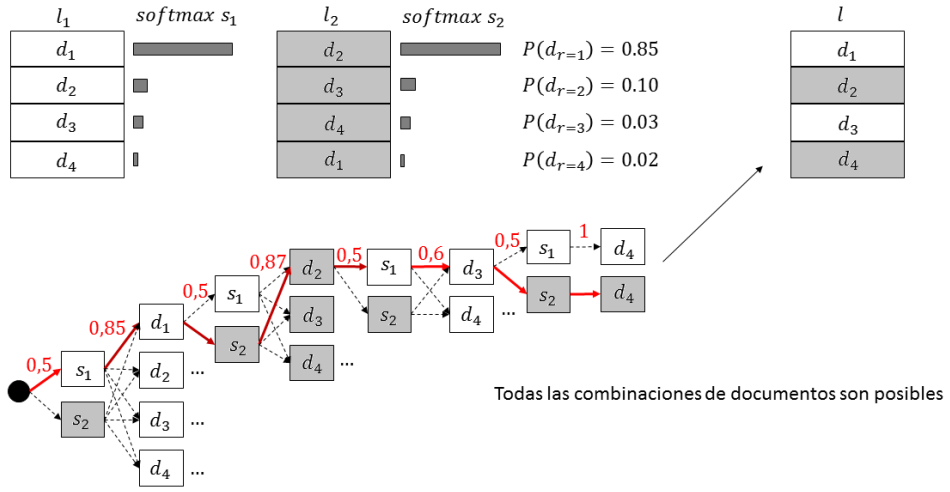

Figura 4. Método probabilístico para intercalado de rankings

2.2.4 Test multivariante

En ocasiones, resulta sencillo generar múltiples variantes de un mismo sistema. Comparar los distintos buscadores dos a dos es muy costoso, y, en función del número de variantes desarrolladas, puede ser imposible de llevar a la práctica una comparación de todos con todos. Para solucionar este problema, aparecen los tests multivariante.

Este tipo de tests son una generalización de los tests A/B mostrados en el apartado anterior. Se diferencian, fundamentalmente, en que ya no se van a comparar simultáneamente dos variantes de un sistema, sino que se compara un número cualquiera, mayor que dos. La principal limitación de estos tests es la necesidad de obtener una cantidad de información mayor para realizar la comparativa entre los buscadores evaluados que la que se necesitaría en caso de combinar solamente dos.

Al igual que los tests A/B, los tests multivariante se pueden llevar a cabo de dos maneras diferenciadas: Redirigir el tráfico del buscador a cada una de las diferentes variantes del sistema, o intercalar los resultados de los diferentes sistemas en un único ranking. Para esta segunda

opción, se han presentado diferentes propuestas, que, en muchas ocasiones, presentan una generalización de los métodos de intercalado para tests A/B. Por ejemplo, Schuth et al. (2014) presentan una generalización del método team draft mostrado en el apartado 2.2.2. para permitir la combinación de más de dos rankings, y Schuth et al. (2015) muestran una modificación del método probabilístico explicado en el apartado 2.2.3.

3. Búsqueda y evaluación

Como se ha explicado en la sección 1.2., el presente trabajo tiene dos objetivos básicos. Por un lado, el desarrollo de una aplicación que permita proporcionar funcionalidades de búsqueda avanzadas sobre un metabuscador, y, por otro lado, la creación de un sistema que posibilite la comparación y evaluación simultánea de diferentes sistemas de búsqueda. Para poder cumplir ambos objetivos, se ha desarrollado una única aplicación que permite ambas cosas. A la hora de construir dicha aplicación, se ha llevado a cabo una selección de herramientas, técnicas y algoritmos tanto de búsqueda como de evaluación para su implementación en el preesente trabajo.

Por ello, en el presente capítulo se detallan aquellos métodos y técnicas de búsqueda y evaluación implementados en la aplicación desarrollada. La estructura del presente capítulo es, por tanto, idéntica a la del capítulo 2: En primer lugar se presentan las decisiones tomadas para el desarrollo de funcionalidades avanzadas de búsqueda (metabúsqueda, gestión de sesión, relevance feedback y diversificación de resultados), y tras ello, se muestran los métodos y algoritmos de evaluación implementados.

3.1 Funcionalidades avanzadas de búsqueda

Como se ha visto en el capítulo anterior, existen múltiples posibilidades para implementar diferentes funcionalidades de búsqueda, como metabúsqueda, gestión de sesión, relevance feedback o diversificación de resultados. En este capítulo, se explican las soluciones adoptadas para la implementación de dichas funcionalidades en la aplicación desarrollada.

3.1.1 Metabúsqueda

Como se ha mencionado en la sección 2.1.1., a la hora de realizar la metabúsqueda es necesario resolver dos problemas diferentes. Por un lado, la selección de los buscadores que se van a agregar, y por otro lado, la combinación de resultados en un único ranking. En esta sección, se explican los algoritmos y decisiones tomadas para la implementación de la metabúsqueda en la aplicación desarrollada.

3.1.1.1 Selección de fuentes

En este proyecto, se han combinado cuatro buscadores distintos, atendiendo a la existencia de una API de búsqueda que permitiese recuperar los documentos de forma sencilla. Los buscadores combinados en este sistema han sido Google, Bing, Faroo y eTools.

- **Google[1]:** Actualmente, es el buscador web más usado. Proporciona APIs de búsqueda personalizadas, tanto en XML como en JSON, mediante su servicio Google Custom Search, que puede apuntar a un único dominio o conjunto de dominios, o a toda la web.
- **Bing[2]:** Anteriormente conocido como MSN Search, es un buscador implementado por Microsoft. Es el segundo motor de búsqueda más popular en Estados Unidos. Bing proporciona una API de búsqueda que permite no sólo buscar páginas web, sino otros recursos como imágenes o videos, así como seleccionar entre los distintos idiomas disponibles en la plataforma.
- **Faroo[3]:** Se trata de un motor de búsqueda basado en P2P, que tiene en cuenta el comportamiento de los usuarios a la hora de devolver rankings. Proporciona una APIs de

[1] http://www.google.com
[2] http://www.bing.com
[3] http://www.faroo.com

búsqueda gratuita, tanto en XML como en JSON, permitiendo escoger entre 3 idiomas distintos (inglés, alemán y chino).

- **Etools Metasearch[4]:** Se trata de un metabuscador que combina más de 15 motores de búsqueda. Si bien es cierto que combina resultados de los 3 buscadores anteriores, pruebas realizadas muestran que hay diferencias notables entre los resultados devueltos por este metabuscador y los rankings de los 3 anteriores. Si bien este buscador no proporciona de forma directa una API, es posible realizar consultas al mismo gracias a la plataforma Carrot2[5].

3.1.1.2 Combinación de resultados

Dado que los buscadores comerciales no proporcionan información sobre los *scores* obtenidos por cada uno de los documentos recuperados, en este proyecto se han empleado métodos basados en posición. En particular, se generan *scores* normalizados a partir de la posición empleando el método Rank-Sim.

Lee (1997) propuso aplicar este método a la metabúsqueda. Rank-Sim trata de dar una mayor importancia a aquellos documentos con posiciones más elevadas en el ranking inicial (y, por tanto, con mayor score en sus buscadores de partida). Este método no solamente permite generar scores para los documentos pertenecientes a cada uno de los rankings, sino que, además, todas las puntuaciones se encuentran en un mismo rango (entre 0 y 1), y normalizadas, por lo que se pueden utilizar directamente para generar el ranking final. Dado un ranking R, El método Rank-Sim emplea la siguiente fórmula:

$$\bar{s}(d) = RankSim(d) = \frac{|R| - rank(d) + 1}{|R|}$$

En esta fórmula, entre dos documentos consecutivos distintos, la diferencia existente es de $1/|R|$, y, en caso de que un documento no figure en el ranking, su valor sería 0. Así por ejemplo, en un ranking de 100 documentos, el documento que apareciese en la décima posición, obtendría 0.91 como *score* normalizado.

Una vez obtenidos estos valores, se puede realizar una combinación lineal ponderada,

$$score(d) = \sum_{i=1}^{|\mathcal{B}|} w_i \, \bar{s}_i(d)$$

donde w_i es un peso asociado a cada ranking, y $w_i \geq 0$. Para evitar que los distintos valores de score obtenidos salgan del rango (0,1), se puede imponer una condición adicional a estos pesos:

$$\sum_{i=1}^{|\mathcal{B}|} w_i = 1$$

Dado que a priori no se dispone de información sobre que buscador de los agregados es mejor, en el metabuscador desarrollado se ponderan los resultados de manera uniforme, es decir, tomaremos $w_i = 1/|\mathcal{B}|$, siendo $|\mathcal{B}|$ el número de buscadores a combinar. Una vez generados, se ordenan los resultados de mayor a menor *score* para formar el ranking unificado, que se mostrará posteriormente al usuario que realice la consulta.

3.1.2 Gestión de sesión y relevance feedback

3.1.2.1 Gestión de sesión

En este proyecto, se ha decidido realizar una gestión de sesión de dos formas muy diferenciadas. Por un lado, se lleva a cabo una gestión de sesión interna a la aplicación, determinando estas

[4] https://www.etools.ch/
[5] http://project.carrot2.org/

sesiones de forma casi transparente al usuario de la aplicación. Por otro lado, se realiza una gestión de sesión explícita, que será controlada por el propio usuario. En esta sección, se explicarán los métodos definidos para realizar ambas tareas.

Gestión de sesión interna

Estas sesiones corresponden con las estudiadas en la sección 2.1.2.1. del presente trabajo. En el contexto de este proyecto, para detectar cuando acaba una sesión de búsqueda y comienza otra, se emplea el método basado en reconocimiento de patrones explicado dicho apartado. Si tomamos las consultas como conjuntos de palabras, mantendremos la nueva consulta en la misma sesión que la consulta anterior si $q_i \cap q_{i+1} \neq \emptyset$. En otro caso, se cierra la sesión, y se crea una nueva, que contendrá la consulta q_i.

Dadas las características de la aplicación desarrollada, la detección de estos conjuntos de consultas ha de realizarse online, en tiempo de ejecución de las consultas. Este método permite llevar a cabo de forma rápida y eficiente la separación entre sesiones. Si bien es cierto que tiene sus limitaciones, estudios previos, como el de Zhang et al. (2013) muestran que aproximadamente un 90% de los cambios de sesión son detectados correctamente por este método.

Gestión de sesión explícita

A diferencia de las sesiones anteriores, las sesiones explícitas, o metasesiones, no almacenan las distintas consultas, sino una serie de documentos seleccionados por el usuario por diversas razones (han considerado que son relevantes, los documentos no se corresponde a aquello que buscan, pero les interesa acceder a ellos más tarde, etc.). Estos documentos pueden ser añadidos por los usuarios desde cualquier búsqueda que realicen. Los usuarios también tienen la posibilidad de borrar resultados de estas sesiones

Estas sesiones explícitas presentan además otra diferencia con respecto a la implementación de las sesiones internas, y es la posibilidad de retomar una sesión antigua. Para ello, los usuarios pueden asignar una etiqueta a la lista de resultados guardados, a través de la cual podrán recuperar los documentos.

3.1.2.2 Relevance feedback

En la sección 2.1.2.3. del presente trabajo, se ha observado que aplicar *pseudo-relevance feedback* sobre una única consulta puede provocar que los términos añadidos a la consulta no sean los términos relevantes. Una forma de evitar o paliar este potencial problema es aplicar *pseudo-relevance feedback* sobre cada una de las consultas que forman una determinada sesión de búsqueda. Es decir, estudiar los k primeros documentos de cada una de estas consultas para tratar de obtener términos relevantes para la necesidad del usuario.

Dada una única consulta, es posible que aquellos resultados que se devuelvan no sean relevantes. Sin embargo, en una sesión de búsqueda, el usuario ha proporcionado más información al sistema sobre aquello que ha estado buscando: ya ha producido diversas variaciones sobre la consulta inicial, lo que podría provocar una mayor aparición de documentos relevantes, que permitiesen identificar aquellos términos más comunes a añadir a la consulta.

Este último enfoque es el que se ha aplicado en este trabajo. Dada una consulta, se ofrece la posibilidad al usuario que la ha realizado de añadir una palabra adicional. Esta palabra adicional a añadir se obtendrá de los *snippets* de cada uno de los documentos devueltos para cada consulta perteneciente a la sesión. A la hora de seleccionar los términos más pertinentes para la nueva consulta, la estrategia empleada es similar a los métodos para el modelo booleano explicado en el apartado 2.1.2.3.: se calculan las frecuencias de aparición de cada uno de los términos, y se añade a la consulta aquella palabra más frecuente que no aparezca en la consulta. Esta decisión se debe a que el modelo seguido por cada uno de los buscadores agregados se desconoce, lo que imposibilita escoger un modelo de manera clara.

3.1.3 Diversidad

Una vez vistos los métodos existentes para realizar la diversificación de los resultados, en este trabajo se ha optado por aplicar un método basado en similitud. Por tanto, en primer lugar, se ha de escoger una función de similitud que indique cómo de distintos son dos documentos, y, por último se escoge un método que diversifique el ranking.

3.1.3.1 Función de similitud

Para definir qué quiere decir que dos documentos sean similares, en este proyecto se ha empleado la similitud por coseno. Se tratan los documentos como vectores formados por los términos que conforman el texto de lo mismos. Las coordenadas de los vectores se calculan mediante el tf-idf (term frequency – inverse document frequency, Salton 1983) de cada término, que se define como:

$$tf - idf(t,d) = tf(t,d) \cdot idf(t)$$

La función tf es una función creciente en términos de la frecuencia del término estudiado en el documento. Permite destacar aquellas palabras que se repitan más veces en el documento en el que aparecen. Por otro lado, idf mide la capacidad de un término para discriminar entre los distintos documentos. Si un término aparece en todos los documentos recuperados, o en la mayoría de ellos, no va a ser un término que permita distinguir un documento frente a otro. Sin embargo, si aparece muchas veces en unos pocos documentos, es muy posible que sea un término representativo de una determinada interpretación de la consulta, y ayuda a diferenciar entre resultados.

Si bien hay muchas posibilidades para calcular estas funciones, en este trabajo se han empleado las siguientes:

$$tf(t,d) = \begin{cases} 1 + \log_2 freq(t,d) & \text{si } freq(t,d) > 0 \\ 0 & \text{si } freq(t,d) = 0 \end{cases}$$

$$idf(t) = 1 + \log_2 \frac{|R|}{|R_t|}$$

con R_t los documentos del ranking en los que aparezca el término t.

Tras obtener los vectores de cada uno de los documentos, calcula la similitud mediante la siguiente fórmula:

$$sim(d_1, d_2) = \cos\left(\overrightarrow{d_1}, \overrightarrow{d_2}\right) = \frac{\overrightarrow{d_1} \cdot \overrightarrow{d_2}}{\left|\overrightarrow{d_1}\right|\left|\overrightarrow{d_2}\right|} = \frac{\sum_t d_{1t} d_{2t}}{\sqrt{\sum_t d_{1t}^2}\sqrt{\sum_t d_{2t}^2}}$$

3.1.3.2 Diversificación del ranking

Previamente, en el apartado 2.1.3.1. se explica el método MMR propuesto por Carbonell et al (1998). En este proyecto, vamos a utilizar un método muy similar a MMR, pero sustituyendo la distancia mínima entre el documento a analizar y los documentos del nuevo ranking por la distancia media. El algoritmo resultante es el siguiente.

1. Se calculan las similitudes entre cada uno de los pares de documentos del ranking inicial.
2. Se selecciona el primer elemento del antiguo ranking, y se coloca como primer elemento del nuevo.
3. Para cada documento no añadido al ranking anteriormente, se calcula el siguiente valor:

$$MR = (1-\lambda) f(d_i, q) - \lambda \frac{1}{|S|} \sum_{d_j \in S} sim\left(d_i, d_j\right)$$

donde S son los documentos añadidos al nuevo ranking, y $f(d_i, q)$ es el *score* del documento i-ésimo en el ranking inicial, R para la consulta q. λ es un parámetro que toma valores entre 0 y 1, y $sim(d_i, d_j)$ es la función de similitud entre documentos.

4. Se escoge el elemento que maximice MR, y se añade al ranking.
5. Se itera desde el paso 3 hasta que no queden más elementos por añadir.

El score de cada uno de los documentos corresponde con el *score* obtenido mediante el método RankSim explicado en el apartado 3.1.1.2., y la función de similitud escogida es la comentada en el apartado 3.1.3.1. Se mantiene el parámetro λ fijo para todas las consultas, con valor de 0.5.

3.2 Evaluación

En esta sección se explican las soluciones adoptadas para la implementación de diversas formas de evaluación. Concretamente, se explica método para realizar tests multivariante, las diferentes métricas empleadas en el trabajo para evaluar buscadores, y . Además, se muestran y analizan los resultados de un pequeño experimento de evaluación llevado a cabo sobre cuatro buscadores comerciales.

3.2.1 Test multivariante probabilístico

A la hora de implementar los métodos de evaluación del proyecto, se ha desarrollado una generalización del modelo probabilístico para test A/B explicado en el apartado 2.2.3. del capítulo anterior. Se ha desarrollado un método de test multivariante para poder comparar simultáneamente los diferentes buscadores que se agreguen al sistema, independientemente de su número. Como se ha explicado anteriormente, es necesario solventar dos problemas para poder realizar la evaluación mediante este método: el intercalado de resultados, y la atribución de clicks.

3.2.1.1 Intercalado de resultados

El algoritmo seguido para el intercalado de resultados es el siguiente. Sea $\mathcal{R} = \{R_1, R_2, R_3, \dots, R_N\}$ el conjunto de N rankings a intercalar. Para cada ranking en $\mathcal{R}$, se define una función softmax. Esta función softmax va a actuar como una distribución de probabilidad, que asignará a cada documento del ranking una probabilidad de ser añadido al ranking unificado. Basándonos en las propuestas de Hoffman et al. (2011) y Hoffman et al. (2013), la función softmax escogida en este proyecto es la siguiente:

$$softmax(d, R_i) = \frac{\frac{1}{rank_i(d)^\tau}}{\sum_{d' \in R_i} \frac{1}{rank_i(d')^\tau}}$$

El parámetro τ presente en la ecuación puede ser modificado para generar diferentes funciones softmax. Representa lo rápido que decaen las probabilidades de seleccionar los documentos de la parte inferior del ranking. Siguiendo las propuesta de Hoffman et al. (2011), y Hoffman et al. (2013), en este proyecto, se ha utilizado $\tau = 3$. En esta ecuación se observa que los documentos con un ranking más elevado van a obtener una probabilidad mayor de ser seleccionados para la parte alta del ranking, dado que esta probabilidad es proporcional al inverso de la posición del documento d en el ranking.

Una vez generadas estas funciones, mientras no se hayan añadido todos los documentos recuperados por los distintos rankings a la lista unificada, se procede de la siguiente forma, muy similar al algoritmo propuesto por Hoffman et al. (2013) para el intercalado de dos buscadores, pero generalizado para intercalar un número N de buscadores:

1. Se escoge un número al azar entre 1 y N. Este número corresponderá con el ranking del que se va a obtener el siguiente documento a incorporar.

2. Se elige un documento de la lista correspondiente. Para ello, se selecciónа un número aleatorio entre 0 y 1, α. El documento seleccionado será el primer documento d_j del ranking que cumpla la siguiente condición:

$$\alpha < \sum_{k=1}^{j} softmax(d_k, R_i)$$

donde i es el ranking escogido en el paso anterior.

3. Se añade el documento escogido al nuevo ranking.
4. Se elimina dicho documento de todas las listas, y se renormalizan las funciones softmax de cada uno de los ranking para que se cumpla que

$$\sum_{d \in R_i} softmax(d, R_i) = 1$$

3.2.1.2 Atribución de clicks

Al igual que en el modelo de intercalado probabilístico, existen dos posibilidades para realizar la atribución de clicks (Hoffman et al. 2011). En primer lugar, asignar cada click al buscador del que fue seleccionado. Por otro lado, calcular el número de clicks esperado dada la lista intercalada.

En este proyecto, se ha desarrollado este segundo método. Para cada consulta, queremos obtener cuántos clicks ha obtenido cada sistema. Para ello, buscamos calcular, para cada sistema, el número de clicks esperado dada la lista intercalada $\mathbb{E}(c_i|l)$. En el método probabilístico, es posible que se obtenga una misma lista intercalada por varios caminos. Esto se muestra, para dos buscadores, en la figura 4. Varios documentos de esta lista podrían venir de buscadores distintos. Sea A el conjunto de las posibles listas de fuentes que podrían dar lugar a la lista intercalada. Tenemos que:

$$\mathbb{E}(c_i|l) = \sum_{a \in A} o(c_i, a) P(a|l, q)$$

donde c_i es el número de clicks asignados al buscador i-ésimo, l es la lista intercalada de resultados, $o(c_i, a)$ es el número de clicks que se atribuirían al buscador i-ésimo en caso de que la lista de fuentes fuese a y $P(a|l, q)$ es la probabilidad de que, dada la lista de resultados l y la consulta q, la lista de fuentes que corresponda a dicha lista sea a.

El método para el cálculo de $o(c_i, a)$ es muy similar a la atribución de clicks para el método team draft: El valor de $o(c_i, a)$ es el número de documentos seleccionados que provienen del ranking i-ésimo acorde a la lista de fuentes a.

Para realizar el cálculo de $P(a|l, q)$, aplicamos la regla de Bayes. De este modo:

$$P(a|l, q) = \frac{P(l|a, q)P(a)}{P(l|q)} = \frac{P(l|a, q)P(a)}{\sum_{a' \in A} P(l|a', q)P(a')}$$

donde

$$P(a) = \frac{1}{|A|}$$

Como $P(a)$ no depende de a, simplificando, se obtiene que:

$$P(a|l, q) = \frac{P(l|a, q)}{\sum_{a' \in A} P(l|a', q)}$$

Ahora, hemos de calcular la probabilidad de obtener la lista l dada la lista de fuentes a. Esta probabilidad se puede calcular como:

$$P(l|a,q) = \prod_{i=1}^{|l|} P(l[i]|a[i], l[1..i-1], q)$$

donde $P(l[i]|a[i], l[1..i-1], q)$ es el valor de la función softmax correspondiente a escoger el documento $l[i]$ de la fuente $a[i]$ dado que se han escogido anteriormente los documentos $l[1], l[2], \dots, l[i-1]$.

A la hora de realizar la implementación, se ha realizado el cálculo mediante un árbol, en el que cada nodo representa a un documento del ranking intercalado, y la fuente de la que procede. Dado un nodo ya estudiado, se procede de la siguiente manera con el siguiente documento en el ranking:

1. Se generan tantos hijos del nodo como fuentes posibles haya para el documento. Cada hijo representará una de las fuentes de las que procede el documento.
2. Se asigna a cada nodo la probabilidad de haber escogido el documento dada la fuente que representa, y los documentos anteriores.

Al final, multiplicando las probabilidades de cada rama del árbol, se obtiene $P(l\,|a,q)$ para cada $a \in A$. En la figura 5 se muestra un ejemplo de la creación de este árbol con dos buscadores.

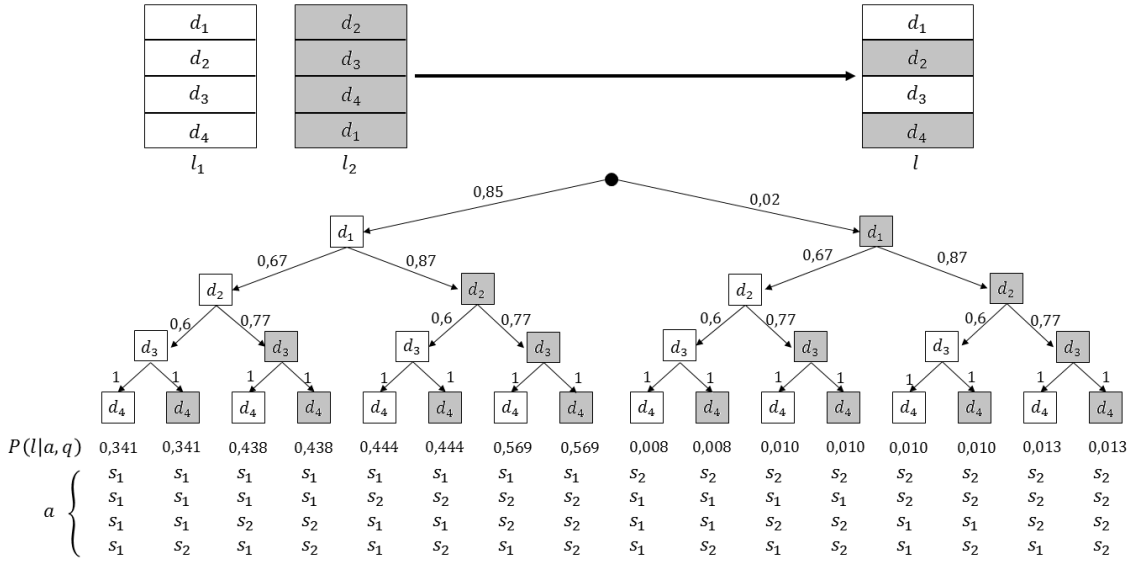

Figura 5. Ejemplo del cálculo de probabilidades para el test probabilístico

Generar este árbol completo es muy costoso cuando se añaden muchos buscadores y el número de resultados es muy elevado. Sin embargo, es fácil ver que basta generar el árbol solamente hasta el último documento seleccionado por el usuario. Dado que habitualmente se seleccionan los primeros resultados del ranking, en la mayoría de ocasiones, esta poda permite mejorar la eficiencia del algoritmo.

Por último, una vez que se ha calculado el número esperado de clicks para cada sistema dada una consulta determinada, se puede obtener obtener cuál de los diferentes sistemas ha obtenido una mayor puntuación para la consulta. De este modo, realizando la comparación para todas las consultas realizadas sobre el sistema de evaluación, se obtiene, para cada buscador, el número de consultas en las que se han seleccionado más resultados que el resto.

3.2.2 Métricas

Una vez puesto a punto el marco de evaluación por intercalado descrito en la sección anterior, la plataforma desarrollada en el presente trabajo incorpora diferentes métricas que se aplican a diferentes componentes integradas en el mismo. Las técnicas de test A/B intercalado

generalmente proponen evaluar los buscadores en términos del número de consultas en las que han recibido más clicks. Complementariamente, es posible no obstante extrapolar los clicks a juicios positivos de relevancia (muy incompletos por supuesto) y a partir de ellos calcular métricas clásicas de recuperación de información (Baeza-Yates et al. 2011). Esta forma de calcular las métricas se realiza pues sobre una simplificación importante (un click es siempre indicación de relevancia, y no se considera relevante lo que no ha recibido clicks), pero cabe esperar que refleje de todos modos las diferencias comparativas de efectividad entre los sistemas evaluados. La plataforma soporta además mediciones orientadas a la diversidad de los resultados. En este apartado, se explican las métricas que hemos considerado en el TFG.

3.2.2.1 Precisión

Esta métrica analiza el ratio de documentos relevantes devueltos por un buscador. Representa la probabilidad de que un documento recuperado en un buscador sea relevante. La fórmula general para una consulta es la siguiente:

$$Precision(q) = \frac{|Relevant \cap Retrieved|}{|Total\ results|}$$

En el sistema implementado, como solamente se recuperan 10 resultados para cada buscador, no es posible calcular esta métrica, por lo que se utiliza una versión más común, enfocada a estudiar los k primeros documentos del ranking, con la siguiente fórmula:

$$Precision@k(q) = P@k(q) = \frac{|Relevant \cap \{d_1, \dots, d_k\}|}{k}$$

donde $d_1, \dots, d_k$ son los k primeros documentos del ranking obtenidos para la consulta.

Esta versión de la métrica estudia, de los k primeros documentos devueltos por un ranking, cuántos de ellos son relevantes, y penaliza la aparición de documentos no relevantes en los primeros resultados.

Este cálculo se promedia sobre todas las consultas realizadas sobre el buscador, por lo que el valor final obtenido será el siguiente:

$$P@k = \frac{1}{|Q|} \sum_{q \in Q} P@k(q)$$

donde Q es el conjunto de consultas a evaluar.

3.2.2.2 Recall

Esta métrica mide qué porcentaje del total de documentos relevantes son devueltos por un buscador. Representa la probabilidad de que un documento relevante sea recuperado por el buscador. La fórmula general para una consulta es la siguiente:

$$Recall(q) = \frac{|Relevant(q) \cap Retrieved(q)|}{|Relevant(q)|}$$

En este proyecto, se calcula una versión enfocada a los k primeros resultados de un ranking, que representa la probabilidad de que un documento relevante cualquiera sea recuperado entre los k primeros resultados de un buscador. La fórmula para esta versión es la siguiente:

$$Recall@k(q) = \frac{|Relevant \cap \{d_1, d_2, \dots, d_k\}|}{|Relevant|}$$

donde $d_1, \dots, d_k$ son los k primeros documentos del ranking obtenidos para la consulta.

El cálculo se promedia sobre todas las consultas, por lo que el valor final obtenido será el siguiente:

$$R@k = \frac{1}{|Q|}\sum_{q \in Q} R@k(q)$$

3.2.2.3 Average Precision

Esta métrica combina precisión y recall. Calculando para cada posición del ranking precisión y recall, y dibujando una gráfica de la precisión en función del recall, esta métrica representa el área bajo dicha gráfica. Para una consulta, se calcula como:

$$AP(q) = avg_{\{k|d_k \in Relevant(q)\}} P@k(q) =$$

$$= \frac{1}{|Relevant(q)|} \sum_{\{k\ |d_k \in Relevant(q)\}} P@k$$

Si un documento relevante no se recupera en el ranking estudiado, se toma ∞ como su posición en el ranking, de tal forma que $P@\infty = 0$.

Agregada sobre varias consultas, esta la métrica toma el nombre de Mean Average Precision, que es el promedio de la métrica anterior sobre todas las consultas que se evalúan. La fórmula para esta métrica es la siguiente:

$$MAP = \frac{1}{|Q|}\sum_{q \in Q} AP(q)$$

3.2.2.4 Reciprocal Rank

Esta métrica da mejor valoración a aquellos sistemas que proporcionan resultados relevantes en las primeras posiciones del ranking frente a aquellos que devuelven documentos relevantes en posiciones muy retrasadas del mismo. La métrica se calcula como el inverso multiplicativo de la primera posición del ranking con un documento relevante:

$$RR(q) = \frac{1}{\min\{k|d_k \in Relevant(q)\}}$$

En caso de que en el ranking no aparezca ningún resultado relevante, la métrica toma por definición el valor 0.

En este proyecto, se va a calcular la métrica conocida como Mean Reciprocal Rank, que se corresponde a la media aritmética de esta métrica sobre todas las consultas estudiadas. Por tanto, la fórmula para esta métrica es la siguiente:

$$MRR = \frac{1}{|Q|}\sum_{q \in Q} RR(q)$$

3.2.2.5 Normalized Cumulative Discounted Gain

Esta métrica estudia la relevancia de los documentos, teniendo en cuenta su posición en el ranking, de tal forma que un documento con mejor posición en el ranking sea más importante que un documento más retrasado, pero de forma más elaborada que la métrica anterior MRR. Además, contempla la consideración de diversos grados de relevancia para los documentos (Järvelin et al. 2002). La fórmula para esta métrica es la siguiente:

$$nDCG(q) = \frac{DCG(q)}{IDCG(q)}$$

donde

$$DCG(q) = \sum_{k} \frac{g(d_k)}{\log_2(1+k)}$$

$$IDCG(q) = \max_{R \in \sigma(D)} DCG(q)$$

En las fórmulas anteriores, R es un ranking, $\sigma(D)$ son todas las posibles reordenaciones de los documentos, y $g(d_k)$ es el grado de relevancia del documento k-ésimo. En este trabajo simplificamos la relevancia al caso binario, es decir:

$$g(d) = \begin{cases} 1 & si\ d \in Relevant \\ 0 & si\ d \notin Relevant \end{cases}$$

De forma similar a precisión y recall, se calcula esta métrica para las k primeras posiciones del ranking, de la siguiente manera:

$$nDCG@k = \frac{DCG@k}{IDCG@k}$$

donde

$$DCG@k(q) = \sum_{i=1}^{k} \frac{g(d_i)}{\log_2(i+1)}$$

$$IDCG@k(q) = \max DCG@k$$

El resultado obtenido para cada búsqueda se promedia sobre todas las consultas realizadas, por lo que se obtendrá lo siguiente:

$$nDCG@k = \frac{1}{|Q|} \sum_{q \in Q} nDCG@k(q)$$

3.2.2.6 Intra List Dissimilarity

Esta métrica trata de valora la diversidad existente entre los diferentes documentos de un buscador, a partir de la similitud entre los distintos documentos. Toma un valor entre 0 y 1, y un valor mayor indica que los documentos devueltos por el sistema son muy diferentes entre sí, mientras que un valor próximo a 0 indica que los documentos son muy similares.

Se calcula, para cada consulta, como:

$$ILD(q) = \frac{2}{|R||R-1|} \sum_{\substack{d_i, d_j \\ i<j}} \left(1 - sim\left(d_i, d_j\right)\right)$$

donde en nuestro caso $sim\left(d_i, d_j\right)$ se define como la similitud por coseno *tf-idf* entre los documentos d_i y d_j. De forma similar a precisión, recall o nDCG, se calcula esta métrica para las k primeras posiciones del ranking, de la siguiente manera:

$$ILD@k(q) = \frac{2}{k(k-1)} \sum_{\substack{d_i, d_j \\ i<j, j \leq k}} \left(1 - sim\left(d_i, d_j\right)\right)$$

En este trabajo, se va a promediar esta métrica sobre todas las consultas realizadas, por lo que el resultado a obtener es el siguiente:

$$ILD@k = \frac{1}{|Q|} \sum_{q \in Q} ILD@k(q)$$

3.2.3 Experimento de evaluación

Aprovechando la aplicación realizada, se ha desarrollado un pequeño experimento de evaluación. Para desarrollar el experimento, se han empleado, en un único ordenador, dos servidores

Gunicorn[6] para permitir el acceso al cliente y al servidor web, un servidor Apache HTTPD[7] para el envío de ficheros estáticos CSS y Javascript, y, por último, bases de datos SQLite3 tanto para el cliente como para el servidor.

3.2.3.1 Resultados

En la Tabla 1 se muestran los resultados del experimento de evaluación realizado. Para este experimento, se han evaluado los buscadores comerciales integrados en el metabuscador, es decir, Bing, Google, Faroo y Etools Metasearch. En la Figura 6 se muestran gráficos comparativos de cada una de las métricas calculadas., junto al valor medio de dichas métricas. Tras analizar los datos, un total de 39 usuarios han accedido al sistema de evaluación para realizar un total de 225 búsquedas. De dichas búsquedas, solamente 125 han recibido clicks por parte de los usuarios.

A la hora de analizar el número de consultas ganadas por cada uno de los motores de búsqueda, solamente se tienen en cuenta aquellas consultas en las que se han producido clicks. En caso de que en una consulta no se haya seleccionado ningún documento, no se ha realizado la asignación de dicha consulta a ningún sistema. En caso de que haya un empate entre diferentes sistemas, se asignará la consulta como ganada a todos los sistemas empatados.

Al realizar el cálculo de las métricas clásicas orientadas a relevancia, los juicios de relevancia se han construido sobre cada par consulta-usuario. Es decir, en caso de que dos usuarios distintos realicen una misma consulta, analizamos las métricas para ambos casos como si se tratasen de consultas diferentes. Además, para las búsquedas se ha considerado que si una búsqueda no recibe ningún click, o no dispone de resultados, el valor para dicha métrica es 0.

3.2.3.2 Análisis de resultados

Tras observar los resultados del experimento, es claro que el buscador agregado que proporciona peores resultados es Faroo, dado que los valores obtenidos para cada una de las métricas analizadas es muy inferior a la de los otros 3 buscadores analizados. Analizando la base de datos de la aplicación, se ha obtenido que, de las 225 consultas realizadas sobre el sistema de evaluación implementado, Faroo solamente devuelve resultados para 134 de ellas, lo que provoca que los valores de las diferentes métricas sean inferiores.

El caso contrario es el de Bing. Este sistema destaca en casi todas las métricas (exceptuando únicamente la métrica ILD@10). Sin embargo, la diferencia entre este buscador y los otros dos buscadores analizados, Google y Etools Metasearch no es tan enorme como la diferencia entre los 3 buscadores y Faroo. De hecho, los valores obtenidos para estos tres sistetmas son muy similares, y en algunos casos la diferencia de valores entre los diversos buscadores aparece en el tercer o el cuarto decimal del valor de la métrica calculada, como ocurre, por ejemplo, en la métrica P@5.

Respecto a las métricas clásicas, Google parece estar algo en desventaja respecto de Etools Metasearch, si bien ha obtenido un número mayor de clicks por consulta y de consultas ganadas que este otro. Esto se puede deber a que Etools es un metabuscador que, entre otros, combina resultados procedentes de Google y de Bing. En caso de que se seleccionen resultados procedentes de dos o más buscadores, la probabilidad de obtener dicho click se reparte entre los distintos sistemas, y en principio, en Etools es más probable que esto ocurra.

Cabe destacar que, principalmente, los resultados relevantes parecen situarse entre los primeros resultados de cada uno de los buscadores. Esto se observa en los valores de las métricas R@5 y R@10. Los valores en cada uno de los sistemas estudiados de R@10 son muy similares a los valores de R@5, lo que permite pensar que no muchos de los resultados seleccionados aparecen a partir de la última posición. Además, como se puede observar en los valores de la métrica MRR, salvo en el caso de Faroo, el primer documento relevante aparece, de media, en la tercera o cuarta posición del ránking.

[6] http://gunicorn.org/
[7] http://httpd.apache.org/

	Clicks esperados /consulta	Consultas ganadas	ILD@10	P@5	P@10	R@5	R@10	nDCG@5	nDCG@10	MAP	MRR
Faroo	0,1017	13	0,4999	0,0204	0,0138	0,0529	0,0654	0,0534	0,0570	0,0451	0,0747
Bing	0,4330	55	0,9264	0,1289	0,0858	0,3057	0,3602	0,2945	0,3133	0,2592	0,3537
Google	0,4305	48	0,9306	0,1271	0,0836	0,2663	0,3103	0,2594	0,2721	0,2198	0,3139
Etools	0,3548	43	0,9499	0,1280	0,0836	0,3019	0,3575	0,2755	0,2932	0,2338	0,3240

Tabla 1. Resultados del experimento de evaluación.

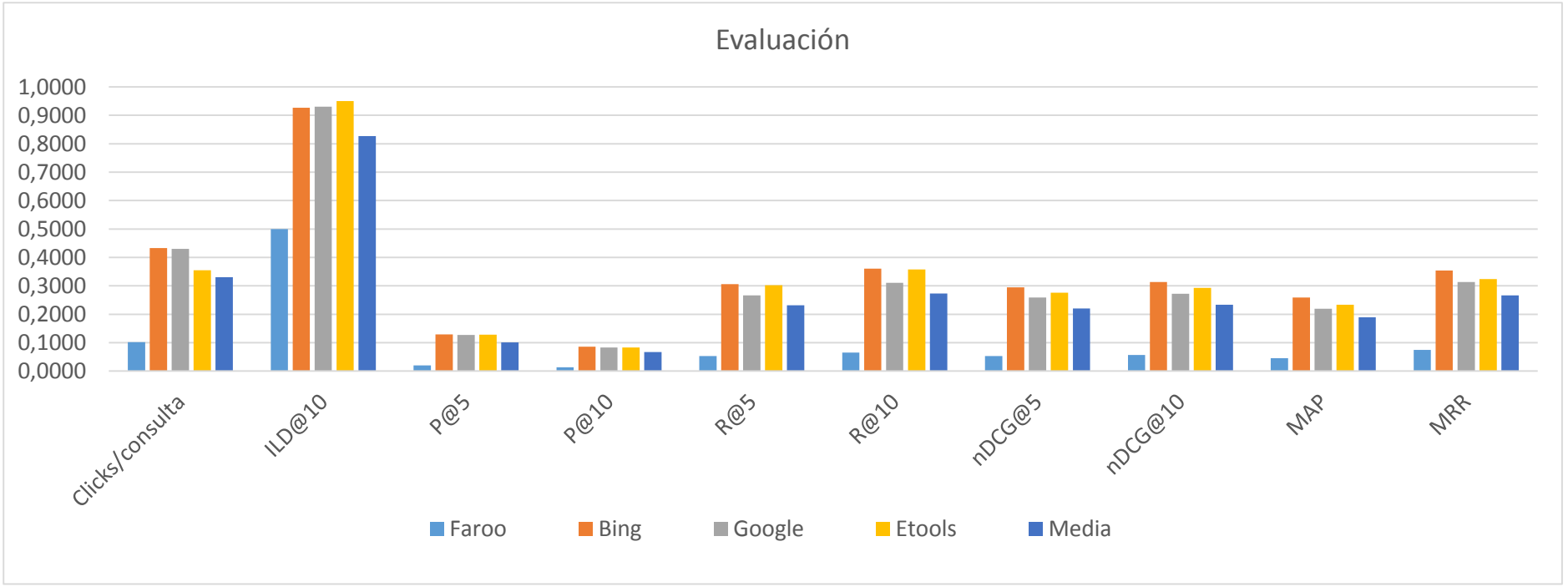

Figura 6. Gráfica resumen de los resultados del experimento de evaluación.

En cuanto a las mediciones de la diversidad de resultados, que se muestran en la métrica ILD@10, excepto Faroo, el resto de sistemas obtiene valores muy elevados. Esto indica que los primeros documentos devueltos por cada uno de los sistemas son muy diferentes entre sí, al menos desde el punto de vista del modelo vectorial de recuperación de información.

3.2.4 Evaluación interactiva

Las funcionalidades de evaluación implementadas en la plataforma permiten evaluar por separado la relevancia y la diversidad de los resultados devueltos por los motores integrados en el buscador. Pero además, permiten evaluar las propias funcionalidades de metabúsqueda y diversificación que la plataforma incorpora. Ello permite comparar diferentes configuraciones de las mismas mediante los parámetros libres que tanto los métodos empleados como su implementación en la plataforma permiten variar.

Los parámetros ajustables incluyen, en el caso de la metabúsqueda, los pesos de los distintos buscadores en la combinación lineal de los scores normalizados, que permiten dar mayor o menor importancia a uno u otro buscador.

En el caso de la diversidad, el peso relativo entre el ranking original y la componente de diversidad. Concretamente, como se ha explicado en la sección 3.1.3., el método implementado se basa en la estrategia avara MMR con la siguiente función objetivo:

$$MR = (1-\lambda) f(d_i, q) - \lambda \frac{1}{|S|} \sum_{d_j \in S} sim(d_i, d_j)$$

donde λ regula la mayor o menor importancia que se da a la posición del documento en el ranking inicial, frente a su valor de diversidad.

Mediante la interfaz desarrollada, la aplicación permite al usuario modificar los valores de todos estos parámetros, y ver dinámicamente cómo varía la disposición de los documentos del ranking acorde a los parámetros modificados. De este modo, comparando los diferentes rankings obtenidos sobre diferentes, es posible tomar una decisión preliminar sobre qué valores se van a utilizar en la aplicación para cada uno de estos parámetros.

4. Arquitectura y detalles de implementación

En este capítulo, se describe la arquitectura definida para la aplicación desarrollada, ZaexSearch, y los modelos de datos diseñados para la de la misma. Además, se muestra la interfaz diseñada para la aplicación, detalles de implementación y de optimización de los algoritmos implementados, y se presentarán las diversas herramientas y librerías utilizadas para el desarrollo de la aplicación.

El sistema desarrollado consta de dos módulos principales: por un lado, un servicio web que se encarga de comunicarse con los diferentes buscadores web agregados a la aplicación, y, por otro lado, un cliente para dicho servicio que utiliza para llevar a cabo la interacción con los usuarios de la aplicación, como se muestra en la Figura 7.

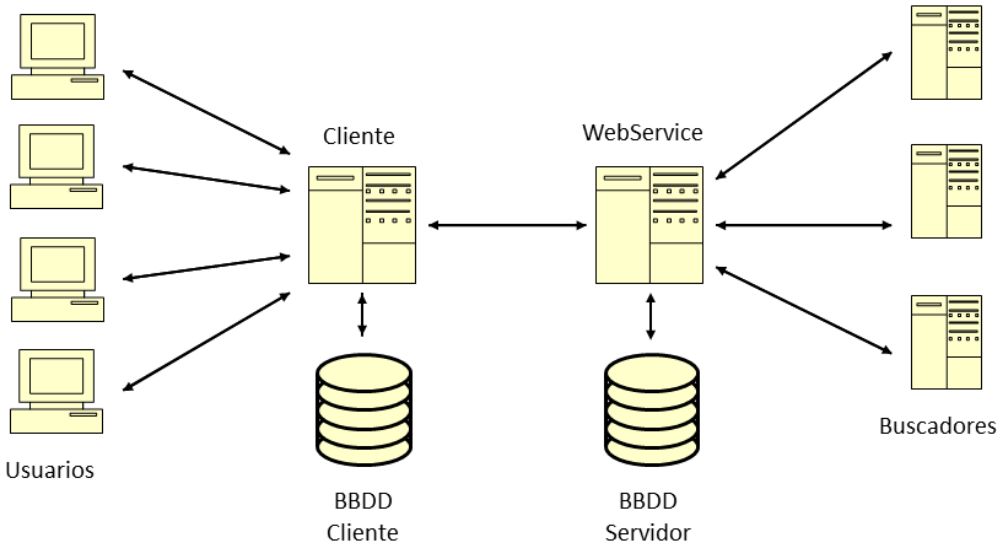

Figura 7. Arquitectura de la aplicación

El servicio web desarrollado puede ser utilizado de manera independiente al cliente, e incluso ser utilizado por diferentes clientes, razón por la que en este proyecto se utilizan dos bases de datos distintas: Por un lado, la base de datos del cliente, y por otro lado, la del servicio web. A continuación, se describen ambos módulos, y sus correspondientes bases de datos.

4.1 Servicio Web

El servicio web desarrollado en esta aplicación es el encargado de proporcionar una interfaz para realizar la comunicación entre el cliente y los diferentes buscadores a agregar. Se trata de un servicio web de tipo REST, en el cual se realizan todas las peticiones por medio de llamadas HTTP, y las respuestas se devuelven mediante JSON. Este servicio web tiene como objetivo realizar las siguientes funciones:

- **Búsqueda y gestión de sesión:** El servicio web se comunica con los diferentes buscadores web agregados en la aplicación para recuperar los documentos seleccionados y agregarlos en un ranking unificado. Simultáneamente a esta búsqueda, el servicio web realiza la gestión de la sesión implícita de los usuarios, es decir, estudiará, a partir de la consulta anterior del

usuario, y de la consulta nueva, si ambas consultas pertenecen a una misma sesión. En caso contrario, creará la nueva sesión.

- **Registro de clicks:** El servicio web registra los *clicks* que se realicen sobre los documentos devueltos. Se registrarán estos *clicks* tanto para metabúsqueda como para tests multivariantes.
- **Diversidad:** Dada una consulta, el servicio web generará y devolverá la matriz de similitudes entre los diferentes documentos recuperados como respuesta a dicha consulta.
- **Relevance Feedback:** El servicio web realiza, a partir de una consulta, y la sesión a la que pertenece, una nueva consulta generada a partir de
- **Intercalado de rankings:** A partir de una serie de buscadores a evaluar, que pueden ser o no ser los rankings agregados para la metabúsqueda, el servicio web ejecuta una consulta sobre todos ellos, y los intercala mediante el método probabilistico explicado en la sección 3.2.1.
- **Evaluación:** Mediante los clicks obtenidos por la aplicación sobre las consultas realizadas sobre la función de intercalado de rankings para evaluación, se obtienen las diferentes métricas comentadas en la sección 3.2.2. para cada uno de los buscadores evaluados.

Las URLs correspondientes para cada función que lleva a cabo el servicio web, los parámetros correspondientes y lo que retorna el servicio web se especifican en el Anexo 1 del presente documento.

El servicio web se divide en varios submódulos, como se puede ver en la Figura 8. A continuación, se proporciona el diseño y una breve explicación de cada módulo. Los diagramas de clase UML correspondientes se muestran en el Anexo 2:

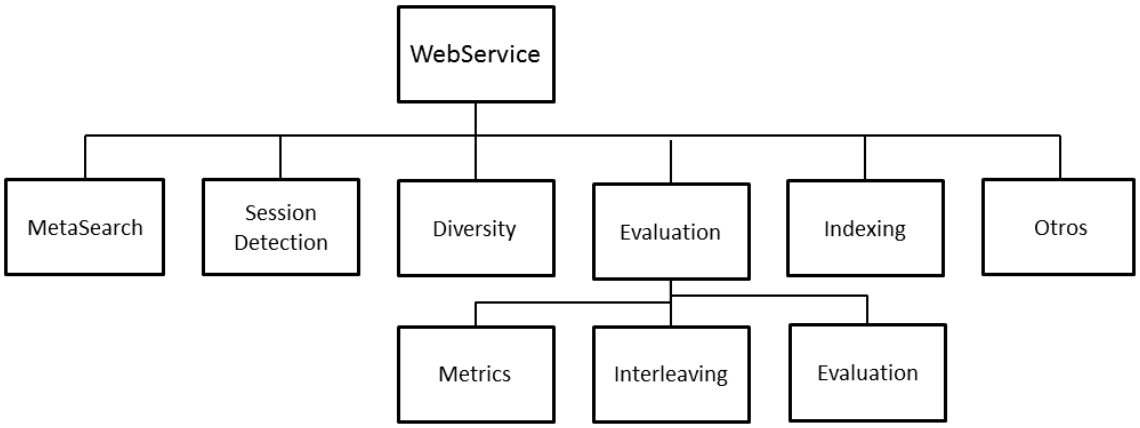

Figura 8. Módulos del servicio web

4.1.1 Base de datos

El servicio web se integra con una base de datos relacional, en la cual se guardan los buscadores a agregar (Google, Bing, Faroo y Etools MetaSearch) y evaluar, las búsquedas realizadas, las sesiones de los diferentes clientes, etc., con el objetivo de emplear esos datos posteriormente para poder aplicar las diferentes funcionalidades mencionadas anteriormente. El diagrama entidad-relación de la base de datos se muestra en la Figura 9.

La entidad Client representa a cada uno de los usuarios de la aplicación. Estos clientes están identificados de manera única por su IP y por el userAgent desde el que se acceda a la aplicación.

Las entidades Session y MetaSession representan los dos tipos de sesiones de los usuarios, implícita y explícita. Ambas sesiones tienen en común el momento de creación de la sesión, y un campo que indica si la sesión está activa o no (active y closed, respectivamente).

La entidad Search representa a cada una de las consultas que se llevan a cabo sobre el sistema. Los resultados correspondientes a cada una de las búsquedas son representados mediante la clase SearchResult, que guarda información variada sobre cada uno de los documentos recuperados

para la consulta, como la URL, el título del documento, su snippet, y, en el caso de metabúsqueda, el valor obtenido de la función de ranking.

La sesión implícita, además de estar compuesta por diferentes búsquedas, almacena también una serie de documentos sobre los que se ha clickado. Esto se refleja en la relación ClickedLink. Esta relación dispone de un campo adicional, que indica si el click realizado va a ser usado o no por el sistema para realizar la evaluación de los distintos buscadores. Por otro lado, las sesiones explícitas guardan una serie de resultados seleccionados por el usuario, lo que se ve en la relación SavedLink.

Es importante destacar la diferencia entre las entidades Ranker y EvaluatedRanker. La primera clase se corresponde con los buscadores que se van a agregar en el metabuscador, y la segunda se corresponde con los buscadores a evaluar. Debido a sus diferencias, aparecen dos relaciones diferentes con la clase SearchResult, que guardan las posiciones de los documentos en los rankings de los buscadores originales. Estas relaciones son, respectivamente, Source e InterleavingSource. En EvaluatedRanker, además del nombre del buscador, se guarda la evaluación del buscador para la última métrica calculada, el numero de búsquedas realizadas, y la métrica ILD del buscador correspondiente.

4.1.2 MetaSearch

Este es el módulo encargado de realizar la comunicación con los servidores de búsqueda, así como de realizar la combinación de los rankings para la metabúsqueda. Para ello, se han definido dos clases abstractas, Searcher y MetaSearcher. La primera clase representa a un buscador particular, y la segunda al metabuscador. El diagrama de clases para este módulo se muestra en la Figura 25.

De la clase Searcher heredan 4 clases, correspondientes a cada uno de los buscadores, que son las clases encargadas de solicitar al sistema correspondiente una lista de resultados.

Por otro lado, de la clase MetaSearcher solamente hereda una clase, que es la que va a combinar los resultados obtenidos por las clases anteriores. Además, la clase MetaSearcher va a ser la que va a llamar a cada uno de las clases que heredan de Searcher, para obtener simultáneamente todas las listas de resultados. Durante la combinación de resultados, las clases que heredan de MetaSearcher registran en la base de datos los diferentes resultados.

Además, se ha implementado una clase, DocSource, que asocia, a cada posición del ranking una lista de fuentes, y una lista de enteros, correspondiente a las posiciones de cada documento en cada fuente.

4.1.3 Session Detection

Se trata del módulo que realiza la identificación y detección de las sesiones implícitas de la aplicación. Para ello, se ha definido una interfaz que permite diversos métodos de detección de sesión.

Todas las clases que implementen dicha interfaz requieren el siguiente método:

- `detectSession(client,query,oldSession)`: Este método decide, a partir de la consulta, el usuario de la base de datos y su anterior sesión, si la nueva consulta pertenece o no a la base de datos. Ha de devolver la sesión antigua, en caso de que la consulta pertenezca a la sesión anterior, y la nueva sesión, en el caso de que no.

En este proyecto, solamente una clase implementa esta interfaz, y se trata de la clase SearchPatternDetector, que emplea el método de detección de sesión basado en patrones léxicos explicado en el apartado 2.1.2.1. El diagrama de clases para este módulo se muestra en la Figura 22.

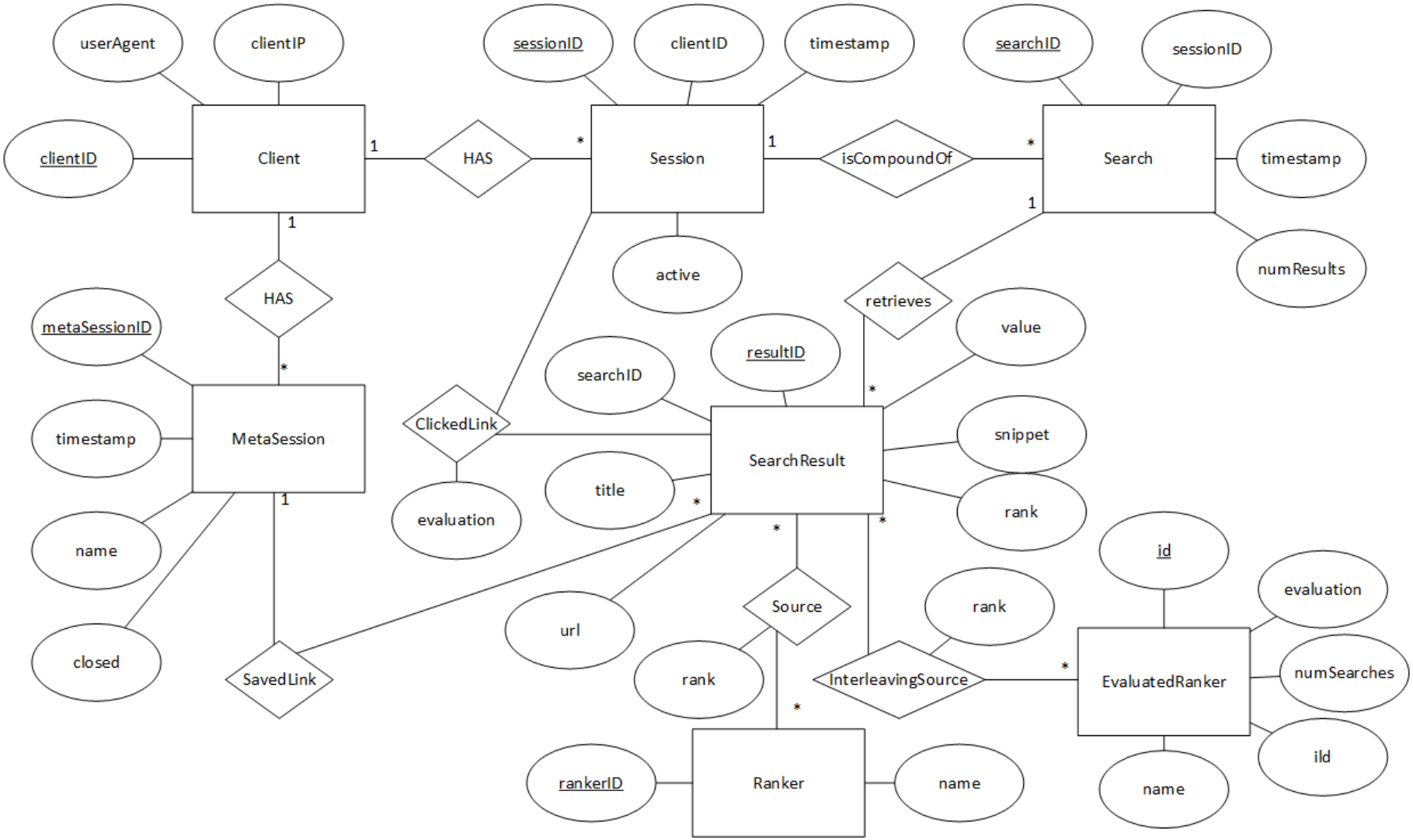

Figura 9. Diagrama ER de la base de datos del servicio web.

4.1.4 Indexing

Este módulo se emplea para crear y gestionar índices, y vectores de documentos. Consta de dos clases. La primera clase, Indexer, se trata de una encapsulación de la librería Pylucene. Pylucene es una librería para Python que permite construir índices, realizar búsquedas sobre ellos, y recorrer los diferentes términos que forman el mismo. Estos índices se van a utilizar para seleccionar términos en relevance feedback, o para hallar similitudes entre documentos. El diagrama de clases para se muestra en la Figura 29.

La segunda clase, FreqVector, representa un vector de documento, y consta de diversos métodos para operar de forma sencilla con él, como el módulo, la suma de vectores o el coseno de dos vectores, que permiten, entre otras funciones, calcular la similitud entre dos documentos.

4.1.5 Diversity

Este módulo es el encargado de generar la matriz de similitud entre los documentos de una consulta, para después diversificar los resultados o evaluar la diversidad entre ellos. Consta de dos clases: La clase DiversityMatrix, que será la encargada de generar la matriz de similitudes, y una clase auxiliar, DocDistance, que representa a la terna formada por dos documentos y la similitud entre ellos. El diagrama de clases para este módulo se muestra en la Figura 23.

Para hallar la similitud entre dos documentos, se emplea la clase FreqVector del módulo Indexing, mediante el cual se generará un índice con las palabras procedentes del ranking que se quiere diversificar.

4.1.6 Evaluation

En este módulo, se encuentran todas aquellas funcionalidades adicionales que tienen que ver con la evaluación de los diferentes sistemas de búsqueda. Tiene como función no solo evaluar los diferentes buscadores, sino también intercalar los diferentes rankings, mediante el método probabilistico explicado anteriormente. Se divide en tres submódulos: Interleaving, Evaluation y Metrics.

4.1.6.1 Interleaving

Este submódulo contiene las clases y métodos encargados del intercalado de resultados. Todas las clases dedicadas al intercalado de documentos, heredan de la clase EvaluationInterleaver (que a su vez hereda de la clase MetaSearcher) y deben implementar dos funciones:

- `doMetaSearch(linkLists : List<SearchResult>, values={})`
- `doMetaSearchNotSaveSource(linkLists:List<SearchResult>, values={})`

que realizan el intercalado de documentos, con una pequeña diferencia. La segunda función no guarda en la tabla Source de la base de datos las fuentes de las que procede cada resultado, para poder separar los buscadores evaluados de los buscadores del sistema en producción. Ambos métodos guardan, sin embargo, los buscadores a evaluar de los que proceden los resultados y la posición de los resultados en los rankings de dichos buscadores en la tabla InterleavingSource.

4.1.6.2 Evaluation

Este submódulo permite llevar a cabo los test multivariantes. Para ello, se ha definido una interfaz, TestAB, que realiza todas las tareas para el test: Intercalado de resultados, gracias a las clases implementadas en Interleaving, y evaluación del sistema. Todas las clases que implementen esta interfaz deben tener los siguientes métodos:

- `interleaveResults(linkLists:List<List<SearchResults>>)`
  Intercala las listas de resultados en una única. Para ello, emplea las clases y métodos del submódulo Interleaving explicado anteriormente. Debe devolver el ranking unificado.
- `evaluateClicks()`
  Evalúa el sistema calculando el número esperado de clicks para el mismo, a partir de los

documentos seleccionados por los usuarios. Debe devolver una lista de buscadores con sus valores de evaluación.

- `evaluateQueries()`
  Evalúa el sistema calculando el número de consultas ganadas por cada buscador, a partir de los clicks esperados para cada buscador en cada consulta. Debe devolver una lista de buscadores con sus valores de evaluación.

En este proyecto, la única clase que va a implementar esta interfaz va a ser ProbabilityABTest, que implementa el método probabilístico explicado en el apartado 3.2.1. Para poder realizar los cálculos correspondientes, se añade una clase auxiliar, ProbBranch, que representa una rama del árbol generado en la evaluación de cada búsqueda. El diagrama de clases para este submódulo y para el módulo de Interleaving se muestra en la Figura 27.

4.1.6.3 Metrics

Este submódulo va a ser el encargado de realizar una evaluación de los buscadores a partir de las diferentes métricas explicadas en la sección 3.2.2. En la aplicación desarrollada, se ha creado una clase abstracta, Metric, que representa una métrica cualquiera, de la que van a heredar las distintas métricas implementadas.

Para ello, deberán implementar una única función:

- `calculate(results:List<SearchResults>,dictClicks:`
  `List<ClickedLinks>)`

que recibe un ranking de resultados, y una lista con los resultados relevantes seleccionados por los usuarios, y devuelve el valor correspondiente de la métrica.

De esta clase derivan múltiples subclases, como PrecissionAtK, NDCGAtK, RecallAtK, MRR, o MAP. Este módulo contiene también la clase ILD, encargada del cálculo de la métrica ILD, pero, como a diferencia de las demás, esta métrica no está relacionada directamente con la relevancia de los documentos, y se trata por tanto por separado. El diagrama de clases se muestra en la Figura 28.

4.1.7 Otras clases y funciones

Además de las clases y módulos presentados anteriormente, en la implementación del servicio web se han empleado algunas otras, encargadas de proporcionar la API del servicio web. Destacan tres tipos:

- **Models:** Se trata de las clases encargadas de representar cada una de las tablas de la base de datos. Además de las entidades explicadas en el apartado 4.1.1., existen clases para ClickedLink, SavedLink, Source, e InterleavingSource. En la Figura 24 se muestran las clases correspondientes para cada entidad en la aplicación desarrollada.
  La gestión de las metasesiones se va a desarrollar desde estas clases. Concretamente, desde la clase MetaSession que implementa métodos para modificar el nombre de la sesión, añadir un enlace a la misma, o borrar un enlace.
- **Views:** Son los diferentes métodos encargados de responder a las peticiones de los usuarios al servicio web. Existe un método para cada una de las posibles peticiones al servicio explicadas en el Anexo 1.
- **Serializers:** Son las clases encargadas de realizar la serialización de las clases de la base de datos, para devolver los datos a través del servicio web. Existen serializadores para todas las clases de documentos (resultados de búsqueda, resultados guardados en sesiones gestionadas por el usuario y resultados seleccionados por el usuario mediante clicks), y para los buscadores evaluados. El diagrama de clases para estos serializadores se muestra en la Figura 26.

4.2 Cliente

El cliente del servicio web va a ser el encargado de realizar la interacción con los usuarios, a través de una interfaz gráfica. Tiene como objetivo realizar las siguientes funciones:

- **Interacción con el usuario:** El cliente muestra la interfaz gráfica de la aplicación, permitiendo al usuario interactuar con la misma de forma similar a los buscadores web existentes en el mercado. A través de estas interacciones, se comunicará con el servicio web mediante peticiones HTTP.
- **Diversificación de resultados:** El cliente reordena los resultados a partir de la matriz de similitudes devuelta por el servicio web, para obtener un nuevo ranking con mayor diversidad, al menos entre los primeros resultados.
- **Gráficos de evaluación:** El cliente genera gráficos comparativos entre los diferentes sistemas, a partir de los métodos de evaluación explicados anteriormente.

El cliente web se divide en varios submódulos, como se puede ver en la Figura 10. A continuación, se proporciona el diseño y una breve explicación de cada módulo. Los diagramas de clases correspondientes se muestran en el Anexo 2: Diagramas UML:

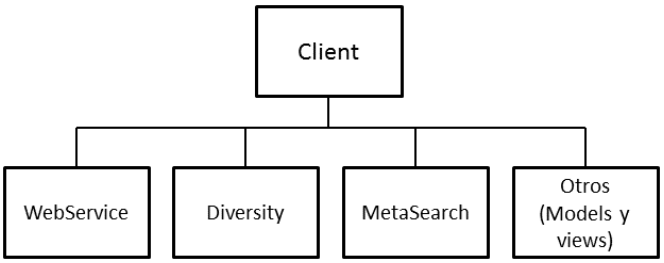

Figura 10. Módulos del cliente web.

4.2.1 Base de datos

El cliente se integra con una base de datos relacional, en la cual se van a registrar algunas de las búsquedas realizadas, o las matrices de similitud. El diagrama entidad-relación de la base de datos se muestra en la Figura 11.

La entidad Search se corresponde con las consultas que se llevan a cabo sobre el sistema. Los resultados que corresponden a cada una de las búsquedas guardadas en la base de datos son representados mediante la clase SearchResult, que guarda información diversa sobre cada uno de los documentos recuperados para la consulta, como la url, el título del documento, su snippet, y, en el caso de metabúsqueda, el valor obtenido de la función de ranking.

La entidad Ranker almacena información sobre cada uno de los buscadores agregados en el sistema. Esta tabla ha de ser idéntica a la tabla Ranker de la base de datos del servicio web. Esto es así, puesto que el servicio web devuelve los identificadores de los buscadores agregados al notificar las fuentes.

Existen, además de las 3 entidades mencionadas anteriormente, 3 relaciones. La primera relación, es la relación existente entre las búsquedas y sus resultados, dado que un resultado de búsqueda puede pertenecer únicamente a una consulta. La segunda, Sources, es una relación entre Ranker y SearchResult, que permite observar las fuentes de las que procede cada uno de los resultados. Por último, la relación Distances permite almacenar en la base de datos las matrices de similitud entre documentos.

4.2.2 WebService

El módulo WebService está formado por una única clase, cuya funcion es la comunicación con el servicio web. Dispone de dos funciones:

- `call_ws(url:String,userAgent:String)`
  Se trata de una función genérica para realizar la comunicación con el servicio web. Se le pasa como argumentos la url del recurso a solicitar, y el agente de usuario del cliente que realiza la petición. Se enviará este agente en la cabecera HTTP de la petición. La función devuelve la respuesta en JSON del Web Service.
- `call_ws_search(query:String,filter:Boolean,IP:String, UserAgent:String)`
  Esta función solicita resultados de búsqueda directamente al servidor. Codifica la consulta y la IP en la URL, introduce el userAgent en la cabecera HTTP, y solicita al servicio que ejecute la búsqueda en los distintos buscadores agregados. Devuelve la respuesta en JSON del servicio.

4.2.3 Diversity

El módulo Diversity del cliente web se va a encargar de realizar la ordenación del ranking a partir de la matriz de similitud proporcionada por el servicio web. Esta formado por una única clase, DiversityRanking, que se llama en dos puntos de la aplicación.

La primera llamada se realizará para reordenar los resultados en el metabuscador principal, mientras que la segunda se llevará a cabo desde el comparador de rankings diversificados y sin diversificar explicado en la sección 3.2.4.

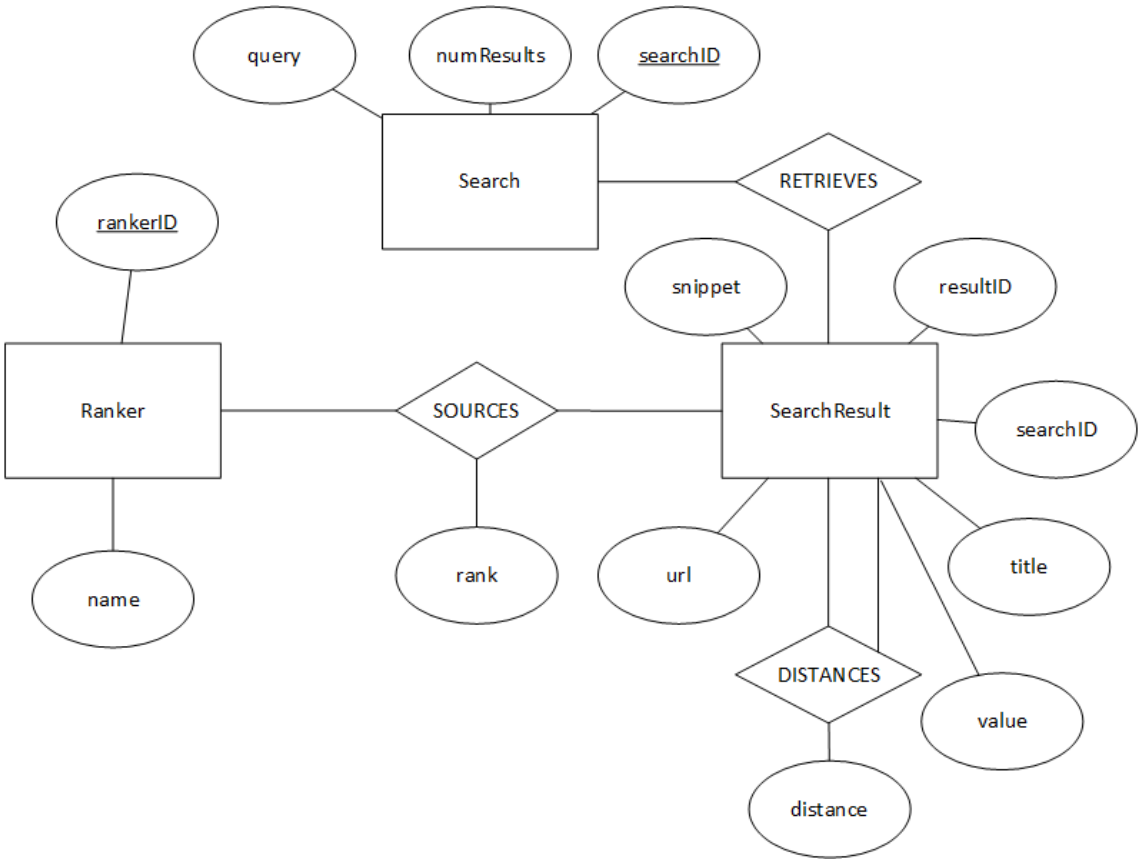

Figura 11. Diagrama Entidad-Relación de la base de datos del cliente web.

4.2.4 MetaSearcher

Para poder realizar la comparación de los rankings dando a cada buscador diferentes valores en la combinación lineal, se ha desarrollado una clase para RankSym, RankSymMetasearcher, similar a la desarrollada para el servicio web.

Esta clase reordenará resultados solamente en esta evaluación. En el resto de ocasiones, será el servicio web el encargado de unificar y ordenar los resultados.

4.2.5 Otras clases y métodos

Además de las clases y métodos anteriores, existen otra serie de clases y funciones empleadas en la implementación del cliente. Destacan dos tipos:

- **Views:** Son las funciones encargadas de proporcionar al usuario las páginas de la interfaz desarrollada, y de comunicarse con las diferentes clases desarrolladas.
- **Models:** Se trata de las clases desarrolladas para representar las diferentes tablas de la base de datos. Se añade además, una clase adicional, Link, para representar los resultados de búsqueda a la hora de mostrar su representación en la interfaz de usuario.

4.3 Optimización

Con el fin de acelerar las diferentes tareas a ejecutar por la aplicación realizada, se han llevado a cabo algunas medidas. En este apartado, se explican los detalles de dichas optimizaciones:

4.3.1 Optimización de las búsquedas

Uno de los cuellos de botella de la aplicación se encuentra en el acceso a los diversos buscadores web agregados. En un buscador web, es común que diferentes usuarios realicen una misma consulta en un breve espacio de tiempo, ya sea por ser una búsqueda muy común, o por un acontecimiento importante. Por eso, se ha desarrollado una caché de respuestas a partir de los resultados almacenados en la base de datos.

Esta caché de respuestas permite, en lugar de solicitar a los distintos motores de búsquedas los resultados para una determinada consulta, obtener dichos resultados directamente desde la base de datos, permitiendo un gran ahorro de tiempo. El sistema implementado solicitará documentos a los buscadores en caso de que no existan datos previos en la base de datos, o en caso de que haya trascurrido más de una hora desde la última vez que se realizó dicha consulta por algún usuario.

Para poder reducir tiempos en la recuperación de resultados de los diferentes buscadores, además, se han paralelizado las solicitudes a cada uno de los buscadores, con el fin de no bloquear el sistema mientras se espera a un buscador. Por último, dado que se han de indexar todos los documentos, se ejecuta este indexado en un hilo separado del hilo de la petición. El indexado de documentos lleva un tiempo no despreciable, y la respuesta al cliente ha de ser lo más inmediata posible, por lo que separar el indexado del hilo principal permite adelantar la devolución de resultados.

4.3.2 Optimización de la diversidad

La obtención de la matriz de similitudes entre los diferentes documentos de una consulta es muy costosa. En caso de calcular todas las similitudes, habría que realizar N^2 operaciones, a lo que se añade el indexado de los documentos para obtener sus correspondientes vectores tf-idf. Por tanto, es necesario tomar medidas para optimizar el cálculo de la matriz. Las medidas tomadas se basan en el hecho de que no es necesario realizar el cálculo de la matriz completa. Para poder obtener todas las similitudes entre documentos, basta con calcular la matriz triangular inferior.

Esto se debe a dos razones. La matriz de similitudes es simétrica, es decir, para cada par de documentos d_i, d_j, se cumple que:

$$sim(d_i, d_j) = sim(d_j, d_i)$$

Además, si se realiza el cálculo de las similitudes mediante la similitud por coseno, como en este proyecto, tenemos que $sim(d_i, d_i) = 1$. Por tanto, en lugar de calcular N^2 similitudes, basta

con calcular $N(N-1)/2$, lo cual, si bien es cierto que sigue siendo $O(N^2)$, conlleva un ahorro de tiempo considerable para un número elevado de documentos.

4.3.3 Optimización de la evaluación

Al realizar la evaluación de las diferentes consultas mediante el método probabilístico explicado en la sección 3.2.1., el cálculo del árbol es costoso (El árbol tendría hasta $|\mathcal{R}|^d$ hojas, donde $\mathcal{R}$ es el conjunto de rankings a combinar y d el número de documentos distintos recuperados por el sistema). Por ejemplo, recuperando 10 documentos idénticos de 4 buscadores distintos (no necesariamente en el mismo orden), el árbol tendría 4^{10} hojas (más de un millón).

Sin embargo, es posible aplicar una mejora al cálculo para evitar realizar tantas operaciones, en caso de que se seleccionen únicamente documentos en la parte alta del ranking. Para realizar los cálculos, basta generar el árbol hasta la posición del ranking en la que se haya realizado el último click. Al no haberse seleccionado documentos en posiciones más bajas del ranking intercalado, las probabilidades de atribuir los clicks a un sistema u otro no se ven modificadas, obteniéndose el mismo resultado.

Para poder realizar el cálculo de la métrica ILD, también se han implementado mecanismos para ahorrar tiempo. En lugar de realizar el cálculo de la métrica bajo solicitud del cliente, al igual que el resto de métricas, se puede aprovechar el hecho de que el valor de la métrica no depende de la interacción del cliente con el sistema, sino que se puede calcular una vez se recuperan los resultados de las diversas fuentes. Por ello, se realiza el indexado de los documentos, el cálculo de la matriz de diversidad y el cálculo de la métrica en un hilo paralelo al de la petición de la consulta de evaluación, de forma similar al hilo que indexa los documentos para relevance feedback en el caso del sistema de metabúsqueda. Esto permite proporcionar de manera inmediata el cálculo de la métrica bajo petición del cliente.

4.4 Interfaz de usuario

En este apartado se describe la interfaz gráfica desarrollada para la interacción de los usuarios con la aplicación, así como la relación entre las diferentes páginas, y su implementación.

Las interfaces web se han diseñado y desarrollado mediante el *toolkit* Google WebStarter Kit[8], que permite el desarrollo de páginas web con un diseño adaptable al tamaño del dispositivo, mediante HTML5, CSS y JavaScript.

4.4.1 Ventana principal

Se trata de la primera pantalla que aparece tras acceder a la aplicación. Es una pantalla muy sencilla, similar a las pantallas de inicio de muchos buscadores comerciales, como Google o Bing. Consta del logo de la aplicación, y de una barra de búsqueda.

Se utiliza esta misma página como pantalla de inicio para las búsquedas en cuatro ocasiones: para acceder a las búsquedas normales, para acceder al sistema de evaluación mediante intercalado de rankings, para acceder al comparador de rankings por diversificación, y para acceder al comparador de rankings por variación de los valores en la combinación lineal para la metabúsqueda. Se puede observar la pantalla en la Figura 12.

[8] https://developers.google.com/web/starter-kit/

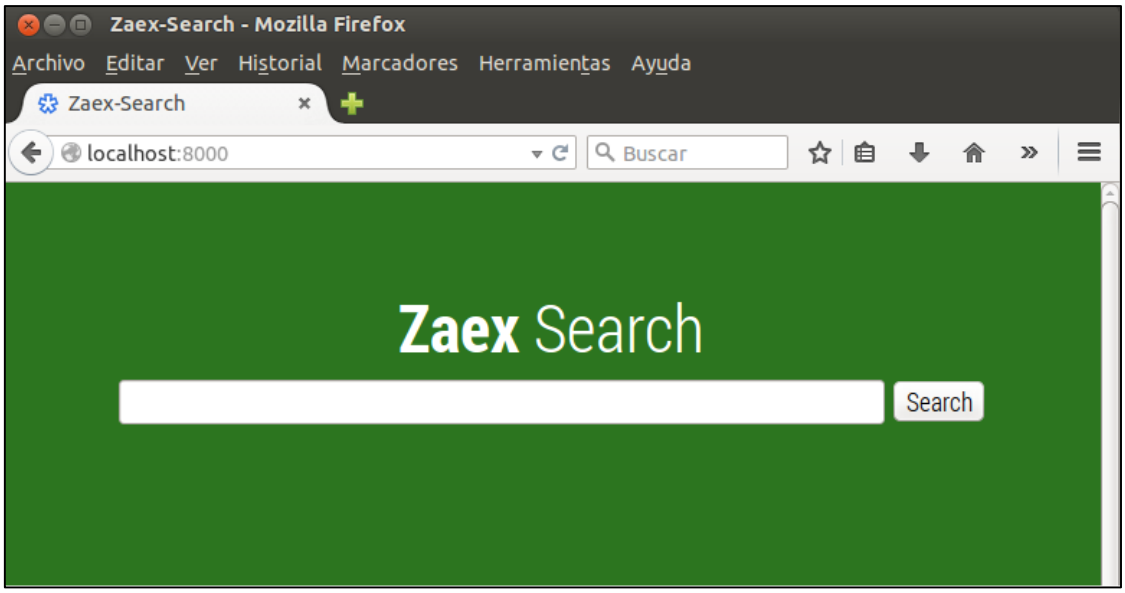

Figura 12. Ventana principal de la aplicación.

4.4.2 Ventana de búsqueda

A esta pantalla se accede tras realizar una consulta sobre el buscador principal de la aplicación. Consta de cuatro paneles: Un panel para mostrar los resultados, un panel para mostrar los resultados que el usuario ha seleccionado en la sesión de búsqueda, un panel para la gestión de las metasesiones, y una página de instrucciones. Todas estas páginas tienen en común un menú en la parte superior de la pantalla, que permite navegar entre los distintos paneles. En caso de que la pantalla sea pequeña se convierte en un menú desplegable, como se puede observar en la Figura 13. En caso de que la pantalla sea lo suficientemente grande, se puede observar en la Figura 14 (1). Además, los distintos paneles comparten también una barra de búsqueda (2). Esta barra de búsqueda permite filtrar los resultados previamente guardados en la sesión gestionada por el usuario actual.

4.4.2.1 Panel de resultados de búsqueda

El panel de resultados se puede observar en la Figura 14. Además de la barra de menú (1) y la barra de búsqueda (2) mencionadas anteriormente, este panel dispone de una lista de resultados (3). Para cada resultado, se muestra su título, la URL en la que se encuentra dicho documento, y una breve descripción del mismo. Además, en la parte inferior izquierda del enlace, se muestran los logos de los buscadores de los que procede cada enlace. Los resultados que se han guardado previamente en la metasesión activa se resaltan de color verde. Si un usuario efectúa un click sobre uno de los resultados, de esta lista, se actualiza el panel de resultados seleccionados, añadiendo este documento.

Junto a la lista de resultados, este panel contiene una barra de herramientas adicional. Esta barra se encuentra situada debajo de la barra de búsqueda, y consta de dos partes: A la izquierda (4), se muestran dos botones. El botón de la izquierda permite al usuario añadir una palabra a la consulta mediante los métodos de relevance feedback explicados en el capítulo 3. El botón de la derecha permite realizar una diversificación de los resultados de la búsqueda.

A la derecha, por otro lado, se encuentra una lista de filtros. Estos filtros permiten mostrar solamente los resultados mostrados por algunos de los buscadores agregados, y ocultar los resultados previamente guardados.

Por último, en la parte inferior de la pantalla, se puede observar un icono de guardado (6). Este icono se emplea para guardar documentos en la sesión gestionada por el usuario. El guardado de documentos se ha implementado mediante mecanismos de Drag&Drop proporcionados por JavaScript. Para ello, basta con arrastrar el enlace hasta el icono de guardado, y soltarlo sobre el

mismo. Una vez se haga esto, se actualiza este panel, resaltando los resultados guardados, y el panel de resultados guardados, que se explicará en breves.

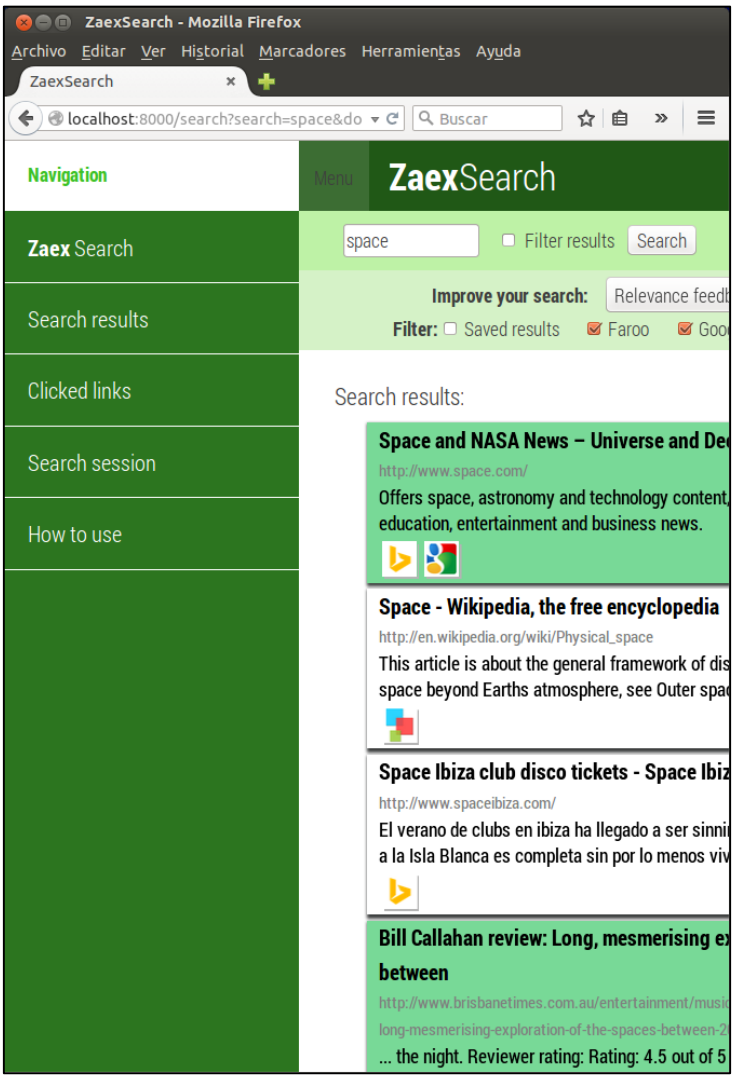

Figura 13. Menú desplegable.

Desde cualquier otro de los paneles de esta ventana, se puede volver a mostrar este panel seleccionando la opción ‘Search results’ en el menú.

4.4.2.2 Panel de resultados seleccionados

En este panel, además de las barras de menú correspondientes, se muestra una lista de resultados seleccionados en la presente sesión de búsqueda (implícita). A diferencia de lo que ocurre en el panel de resultados de búsqueda, en este panel no se muestra ni la procedencia del documento, ni se resaltan los enlaces en caso de que se haya guardado.

Desde cualquiera de los paneles de esta ventana, se puede mostrar de nuevo este panel seleccionando la opción ‘Clicked results’ en el menú. Se puede observar en la Figura 15.

4.4.2.3 Panel de metasesión

Este panel sirve para que el usuario elimine documentos de la sesión interna, guarde la sesión interna proporcionandola un nombre, o abra una sesión guardada previamente. Para ello, se ha dotado a la interfaz de una nueva barra, como se puede ver en la Figura 17 (1). En esta barra, aparecen dos cuadros de texto y dos botones. El cuadro de texto y el botón en la parte superior de la barra permiten al usuario abrir una sesión guardada previamente, a partir del nombre que se

proporcionó a la misma. Por otro lado, el cuadro de texto y el botón de la parte superior de la barra permiten al usuario guardar y cerrar la sesión actual. En caso de que el usuario no seleccione el checkbox correspondiente, el sistema comprobará que no se le ha proporcionado el nombre con el que se va a guardar la sesión previamente a realizar el guardado. En caso de que así sea, no se guardará la sesión. En caso de que el checkbox esté seleccionado, se sobreescribirá dicha sesión en caso de existir.

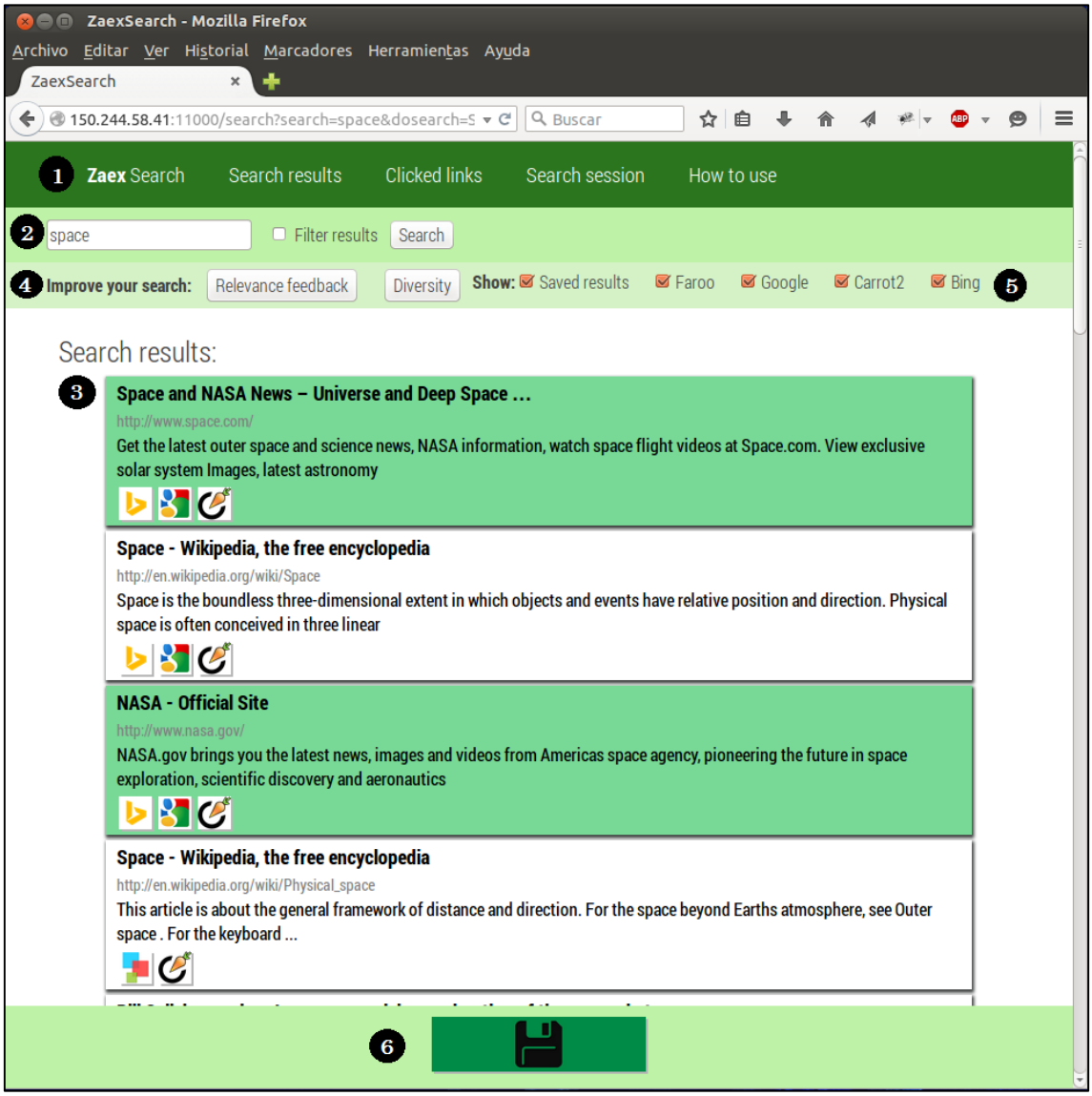

Figura 14. Panel de resultados de búsqueda.

Debajo de dicha barra, se mostrará la lista de documentos guardados (2), con un formato similar al exhibido en el panel de resultados seleccionados. Por último, para realizar la eliminación de documentos de la sesión de búsqueda, se ha implementado un sistema Drag&Drop similar al del guardado de resultados, diferenciándose en el icono de la parte inferior de la pantalla: En este caso, aparece un cubo de basura en lugar de un icono de guardado. Tras borrar un documento, guardar o abrir una sesión, se actualizarán este panel y el panel de resultados de búsqueda.

Desde cualquiera de los otros paneles, se puede mostrar este seleccionando la opción ‘Search session’ en el menú.

4.4.2.4 Panel de ayuda

Este último panel se trata de un manual de ayuda para los usuarios de la aplicación y explica las características del buscador. Se muestra en la Figura 16 y se puede acceder desde la opción 'How to use' del menú.

4.4.3 Ventana de evaluación de resultados

Este es la ventana que muestra los resultados intercalados para la evaluación. Está formada por un único panel, que se puede observar en la Figura 19. Consta de dos partes diferenciadas.

En la parte superior de la pantalla, se muestra una barra de búsqueda (1), que permite realizar una nueva búsqueda sobre el sistema de evaluación.

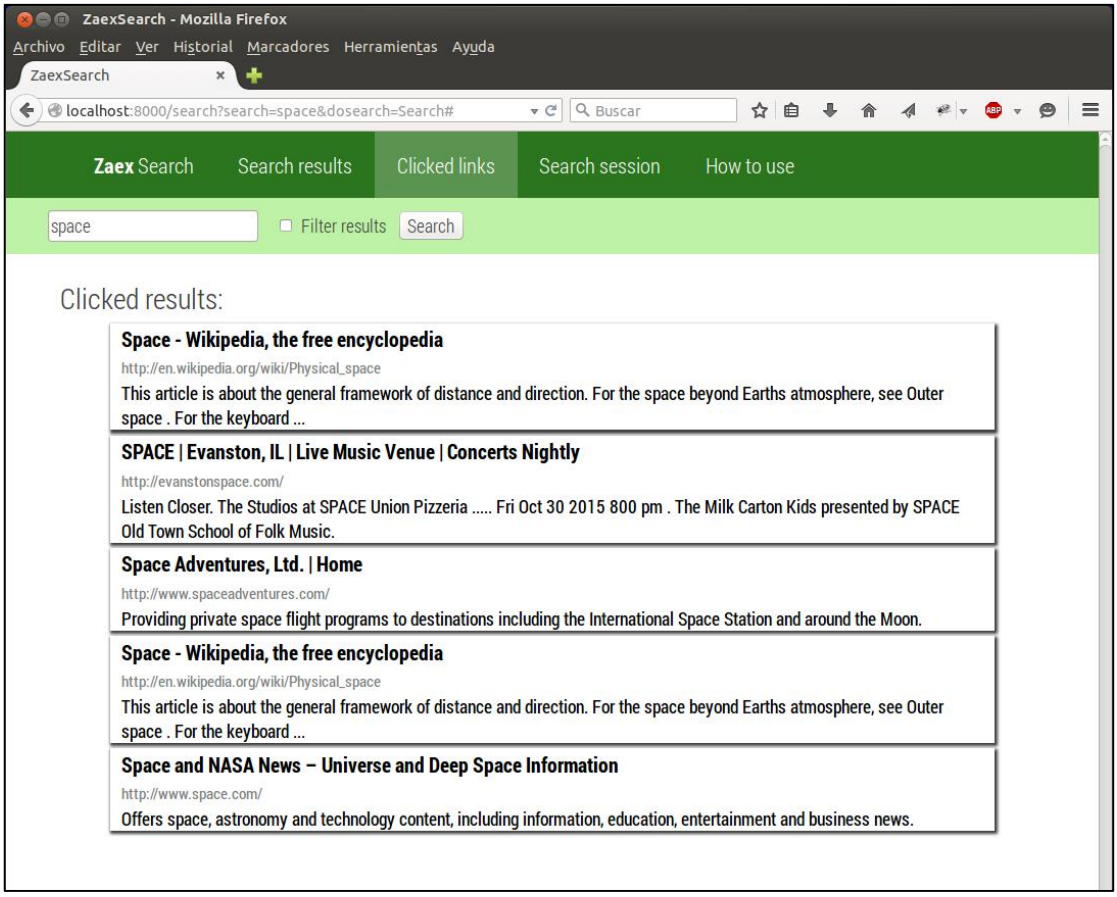

Figura 15. Panel de resultados seleccionados.

Por otro lado, debajo de dicha barra, se muestra la lista de resultados. Cada resultado muestra únicamente el título, la URL y una breve descripción del resultado. Con el fin de no sesgar los resultados de la evaluación en función del buscador, no se muestran las fuentes de las que proceden los distintos resultados. Al seleccionar un documento mediante un click, el sistema registra la interacción, y la enviará al servicio web para su posterior registro.

4.4.4 Ventana de evaluación de la diversidad

Esta ventana va a permitir realizar la comparación entre un ranking sin diversificar y un ranking diversificado, modificando el parámetro λ mostrado en la fórmula para diversificación de resultados de la sección 3.3.2. Esta ventana se muestra en la Figura 20.

En la parte superior de la pantalla, esta ventana consta de una barra de búsqueda similar a la de la ventana de evaluación (1). En esta ventana se muestran dos rankings. Estas dos listas de resultados van precedidas del valor de la métrica ILD para los 10 primeros resultados de cada lista. Esto permite al usuario estudiar la evolución del valor para distintos valores del parámetro

λ. La primera lista de resultados, situada a la izquierda de la pantalla (2), representa el ranking sin diversificar, mientras que la lista situada a la derecha de la pantalla (3), representa el ranking diversificado.

Para variar el parámetro λ, se ha situado un *slider* (4) debajo de la barra de búsqueda, que permite a dicho parámetro tomar valores entre 0 y 1. Cada vez que se modifique la posición de este *slider*, se refrescará el ranking de la derecha, y su métrica, mostrando el nuevo ranking diversificado. Inicialmente, ambos rankings son idénticos ($\lambda = 0$).

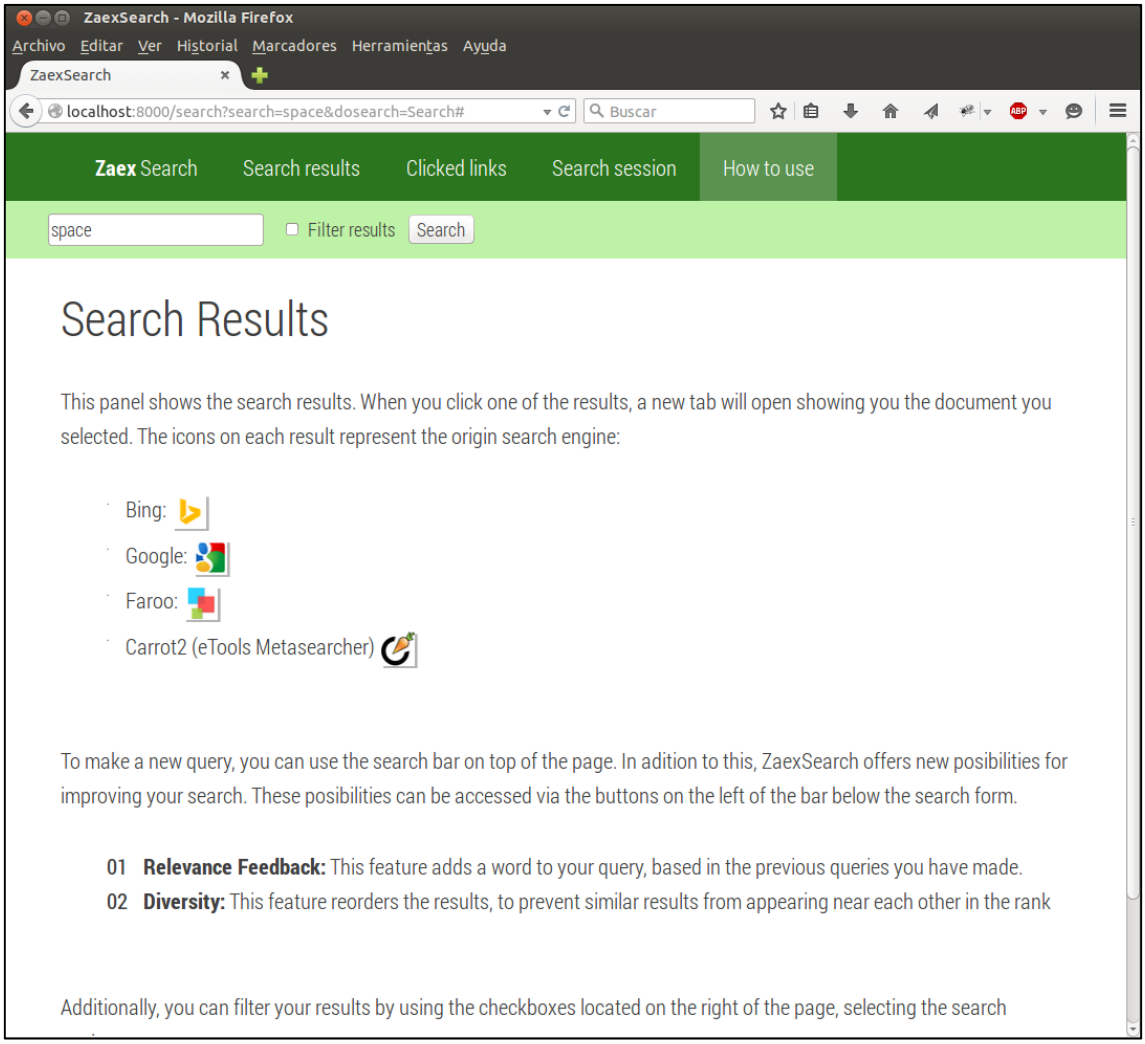

Figura 16. Panel de ayuda.

4.4.5 Ventana de evaluación de la metabúsqueda

Esta ventana permite modificar la importancia de cada buscador en la combinación lineal de los *scores* normalizados por Rank-Sim. Esta ventana se puede observar en la Figura 21.

En la parte superior, se presenta una barra de búsqueda (1), que permite realizar una nueva búsqueda contra este sistema. En la parte inferior de la pantalla (2), se muestra la lista de resultados correspondiente a la búsqueda. Estos resultados incluyen el título, la URL y los *snippets*, además de las fuentes de cada resultado, lo que permite al usuario observar la evolución del ranking al darle más importancia a uno u otro buscador.

Para poder modificar dicha importancia, aparecen los cuatro *sliders* situados en debajo de la barra de búsqueda (3). Todos estos *sliders* toman valores entre 0.001 y 1, y, al modificar

cualquiera de ellos, provocan que se refresque la pantalla, con el ranking actualizado según los pesos correspondientes.

4.4.6 Ventana de informe de evaluación

Esta ventana permite visualizar los resultados de la evaluación de los diferentes sistemas evaluados. Esta ventana se puede observar en la Figura 18.

Para cada métrica calculada, se muestra una tabla (1) con los valores de la métrica para cada uno de los sistemas a evaluar. Además, debajo de dicha tabla, se muestra una representación gráfica (2) que permite comparar de forma rápida los resultados obtenidos. En el caso del número esperado de clicks, y de las consultas ganadas por los buscadores, se mostrará un gráfico circular, mostrando el porcentaje de clicks/consultas que se lleva cada sistema. Por otro lado, en el resto de métricas se mostrará un diagrama de barras comparando los diferentes sistemas.

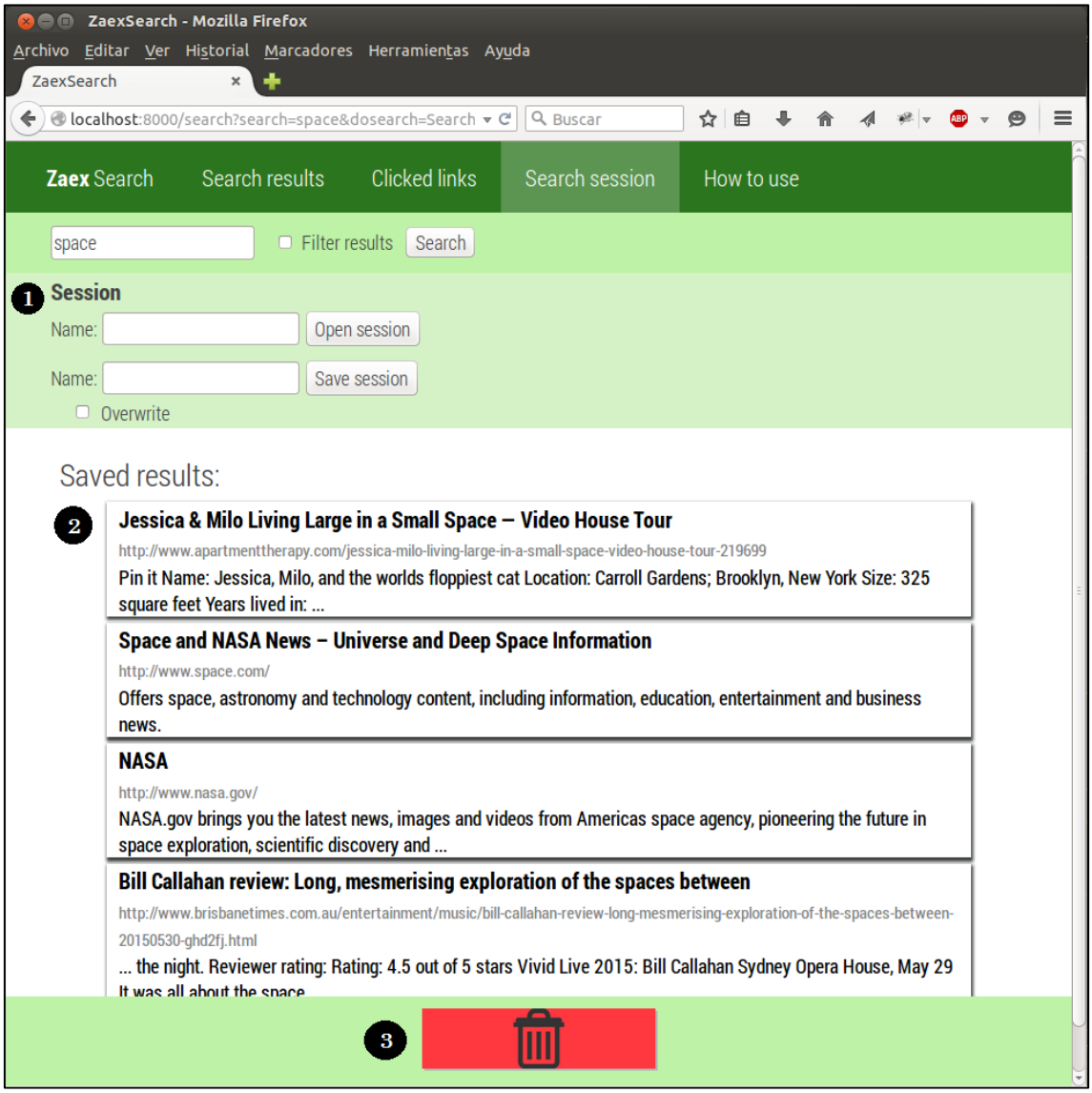

Figura 17. Panel de metasesión.

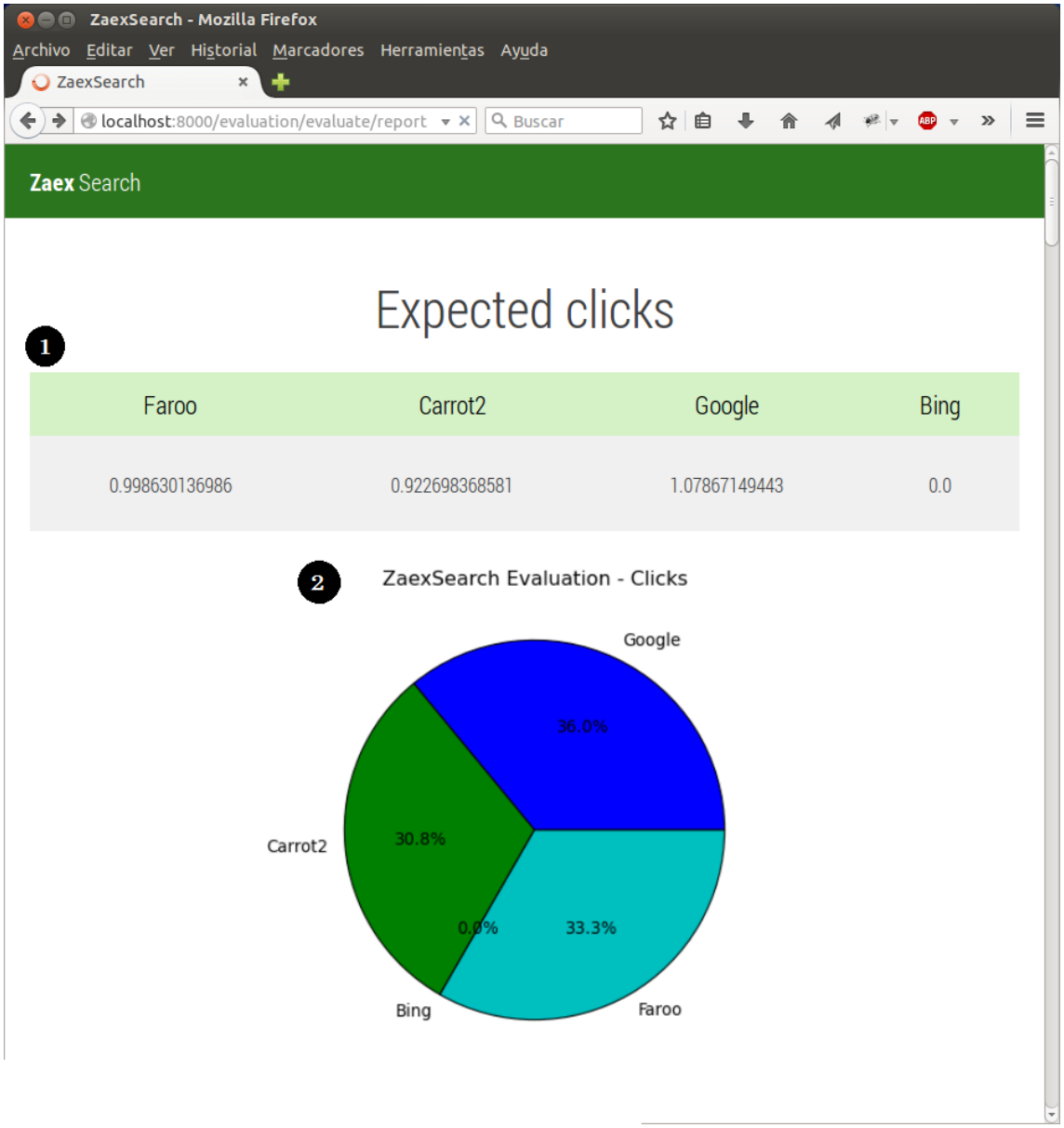

Figura 18. Ventana de informe de evaluación.

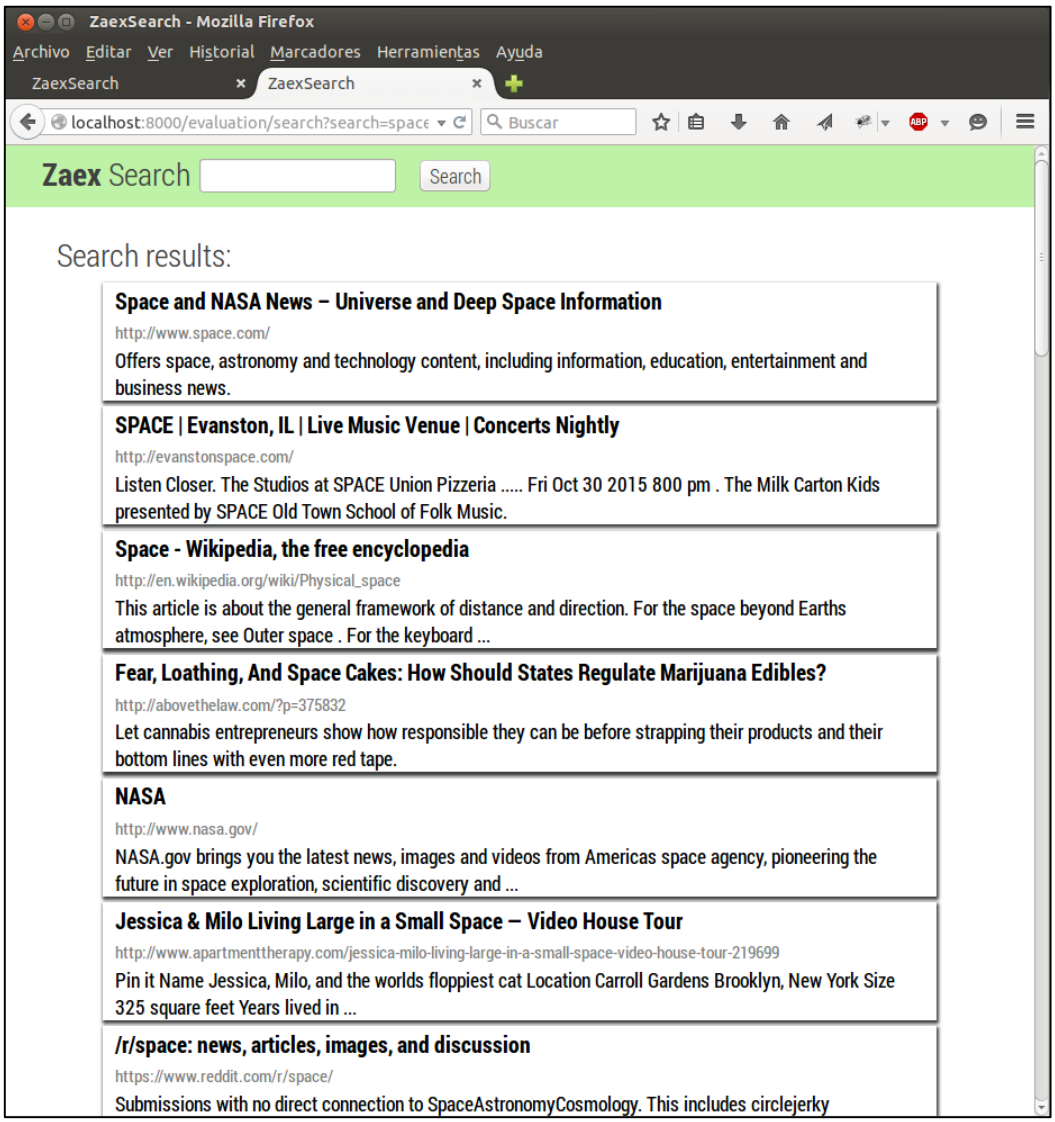

Figura 19. Ventana de evaluación de resultados.

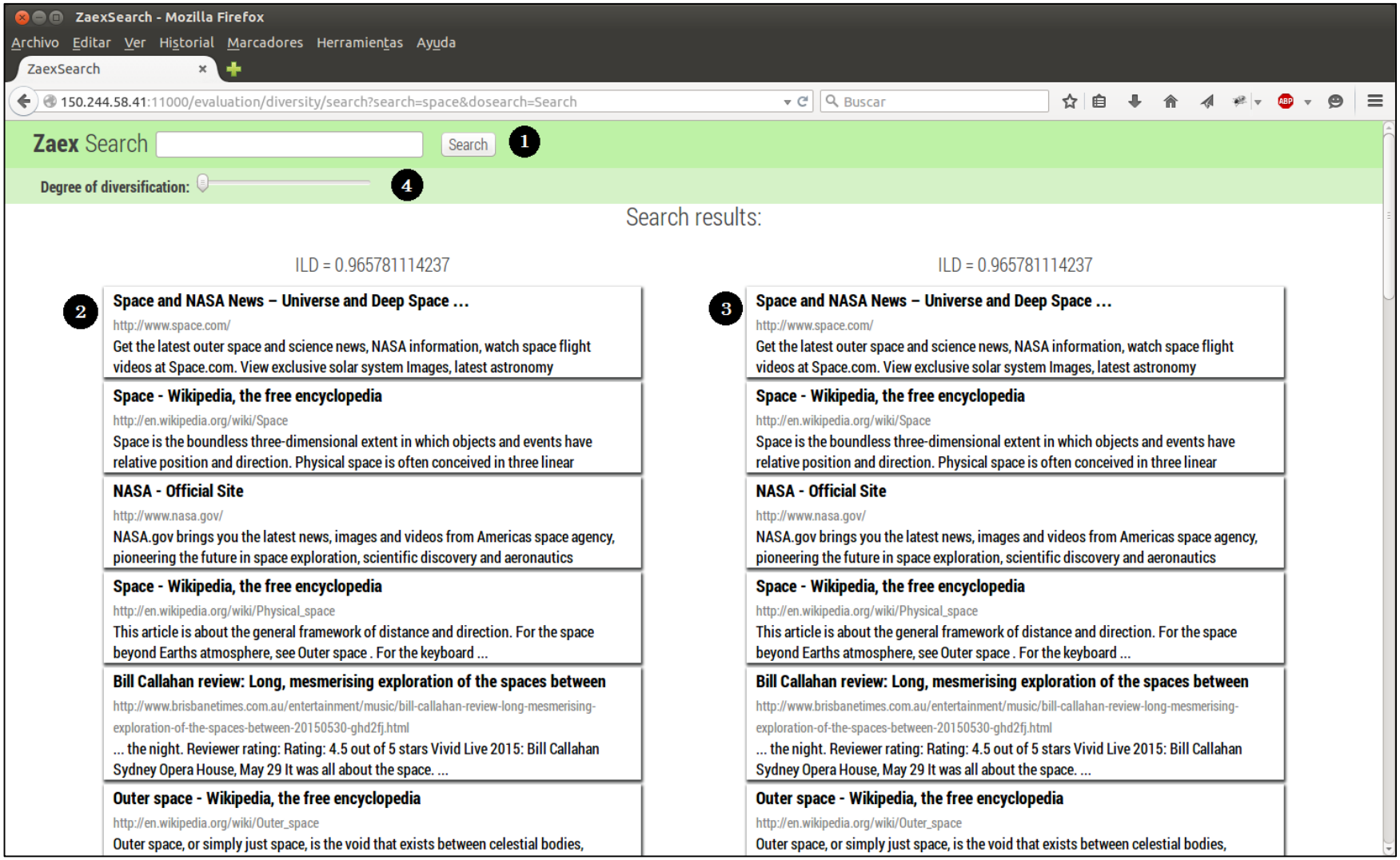

Figura 20. Ventana de evaluación de la diversidad.

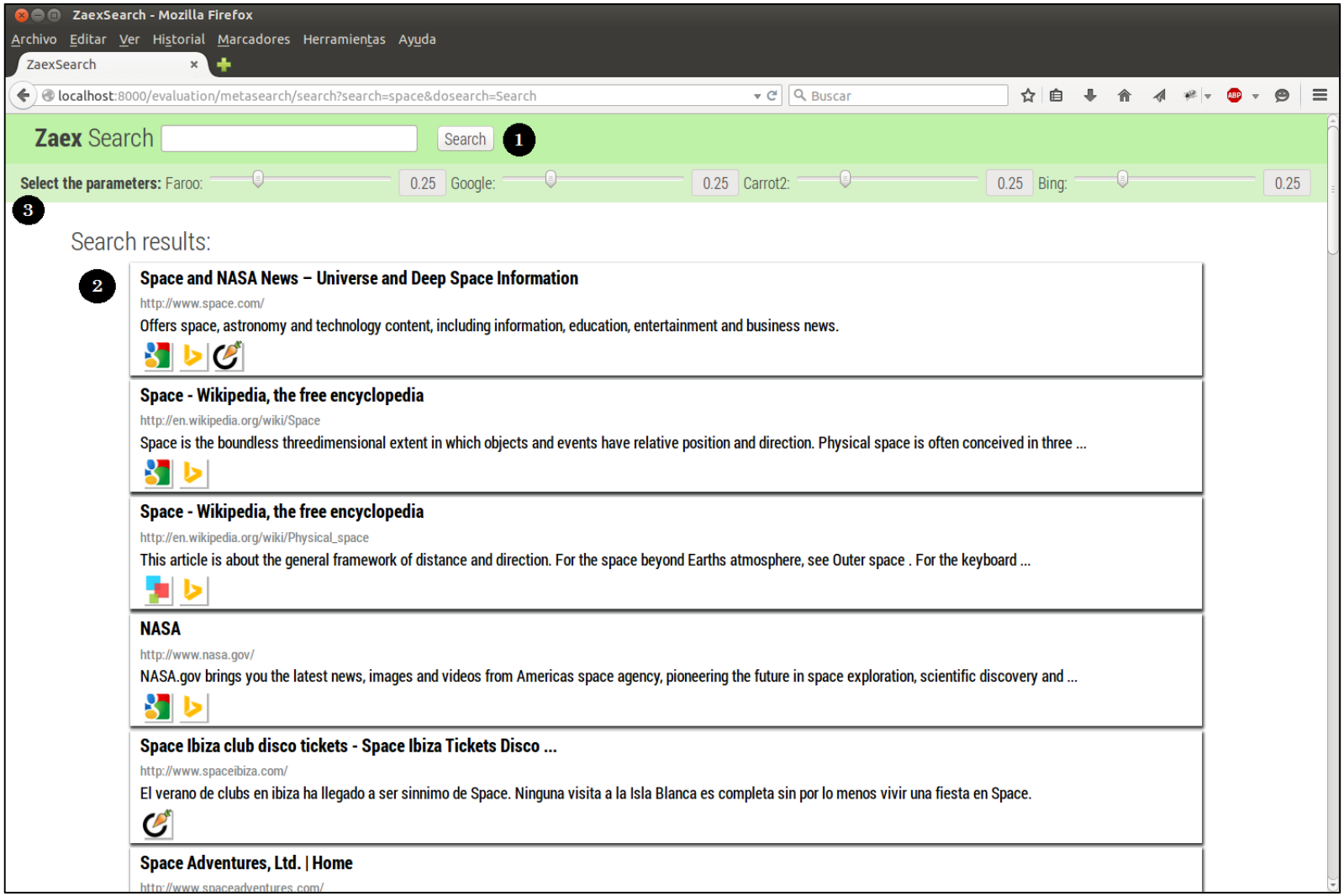

Figura 21. Ventana de evaluación de la metabúsqueda.

4.5 Librerías y herramientas utilizadas

La aplicación se ha desarrollado se ha empleando, en su mayor parte, el lenguaje de programación Python[9], utilizando las siguientes librerías:

- **Django[10]:** Framework para el desarrollo de aplicaciones web sobre Python.
- **Django REST Framework[11]:** Framework para desarrollar servicios web de tipo REST que trabaja sobre Django.
- **Beautiful Soup[12]:** Parser HTML; que permite limpiar las etiquetas HTML del contenido de una página web.
- **Pylucene[13]:** Se trata de una extensión de Python que permite utilizar la librería Lucene[14], desarrollada para el lenguaje de programación Java. Se trata de una librería que permite la creación de índices a partir de textos, así como la búsqueda de los documentos indexados.
- **Matplotlib[15]:** Esta librería permite dibujar gráficos 2D en Python de manera sencilla.

Además de Python, para realizar la interacción entre los usuarios de la aplicación y el cliente web, se ha empleado el lenguaje JavaScript.

Para poder acceder a los resultados del buscador eTools, se ha empleado además la herramienta Carrot2[16]. Carrot2 es una herramienta que, tras realizar una búsqueda, crea diferentes clústers de documentos. Proporciona una API a través de un servicio web REST que se ha utilizado para recuperar los documentos correspondientes.

[9] https://www.python.org/
[10] https://www.djangoproject.com/
[11] http://www.django-rest-framework.org/
[12] http://www.crummy.com/software/BeautifulSoup/
[13] http://lucene.apache.org/pylucene/
[14] http://lucene.apache.org/
[15] http://matplotlib.org/
[16] http://project.carrot2.org/

5. Conclusiones

Concluimos en este capítulo el cuerpo de la memoria con un resumen del trabajo realizado, destacando las contribuciones más importantes, así como las posibles opciones de ampliación y mejora del mismo.

5.1 Resumen y contribuciones

En este trabajo de fin de grado se ha diseñado y desarrollado una aplicación web, a la que hemos dado el nombre de ZaexSearch, que permite, por un lado, agregar diferentes motores de búsqueda para proporcionar un ranking unificado, y, sobre ese ranking, proporcionar diferentes funcionalidades a los usuarios con el fin de mejorar su experiencia, y, por otro lado, comparar diferentes sistemas mediante métodos de intercalado de resultados, que permiten realizar una evaluación de los buscadores a partir de los clicks realizados por los usuarios, sin que estos perciban que están utilizando una versión de prueba del sistema y están siendo expuestos a diferentes alternativas en el subsistema de procesamiento de consultas.

Con el fin de desarrollar el metabuscador, se han investigado y estudiado las diferentes técnicas y herramientas existentes actualmente, para proporcionar una visión global de las mismas. En concreto, se han estudiado a) técnicas de metabúsqueda, para poder realizar la combinación de los motores de búsqueda, b) diferentes métodos para realizar la gestión de sesiones de búsqueda, c) diferentes técnicas existentes para añadir palabras a una consulta a partir del *feedback* implícito proporcionado por el usuario, y d) varios algoritmos para la diversificación de resultados. De todas las técnicas y soluciones documentadas en la literatura para cada una de estas funcionalidades, hemos seleccionado varias para ser implementadas en el metabuscador desarrollado para el presente trabajo.

En el caso de la metabúsqueda, para cada uno de los rankings agregados, se han calculado scores normalizados para cada uno de los documentos recuperados mediante el método Rank-Sim. Una vez hecho esto, se ha realizado una combinación lineal de dichas puntuaciones para obtener un score definitivo para cada resultado, y, a partir del mismo, realizar la ordenación de los resultados.

Para poder aplicar a las búsquedas realizadas métodos de relevance feedback, se ha desarrollado un método novedoso, que añade a una consulta la palabra más común en los documentos recuperados en las diferentes consultas pertenecientes a una sesión de búsqueda. Para poder diferenciar los documentos que pertenecen a cada una de las consultas, se ha optado por un algoritmo basado en patrones léxicos, que establece la existencia de diferentes sesiones en función de las palabras comunes entre consultas consecutivas.

Por último, para realizar la diversificación de resultados, se ha implementado un método basado en la similitud de los documentos, que reordena los diferentes resultados teniendo en cuenta el score obtenido por el documento en el ranking original, y la diferencia del mismo con el resto de documentos.

En cuanto a la evaluación, se ha desarrollado un nuevo método de evaluación, que permite comparar múltiples sistemas, a partir de una generalización del intercalado de ranking propuesto por Hoffman et al. (2011) para dos sistemas. Este método permite realizar una atribución de clicks a cada uno de los motores de búsqueda agregados, con el fin de determinar qué sistema proporciona mejores resultados. Además, con la ayuda de la interfaz desarrollada, se han desarrollado métodos para realizar una evaluación preliminar de los parámetros empleados en los algoritmos de metabúsqueda y diversificación de resultados, para poder decidir, mediante la observación de los ránkings, cuales son los valores más adecuados.

Mediante la aplicación realizada, se ha llevado a cabo un pequeño experimento de evaluación, comparando los buscadores Google, Bing, Faroo y Etools MetaSearch. Tras analizar los resultados, se ha obtenido que, si bien Google, Bing y Etools obtienen resultados muy parecidos para diferentes métricas, Bing destaca en todas ellas, salvo en la métrica ILD@10, donde Etools obtiene los mejores resultados. A tenor de los resultados obtenidos, Faroo obtiene los peores valores para todas las métricas estudiadas.

5.2 Trabajo futuro

El presente trabajo presenta múltiples vías de continuación. A continuación, se detallan algunas de ellas:

Una posible vía de ampliación del presente trabajo es la creación de un índice propio de documentos, y la construcción sobre el mismo de un motor de búsqueda que se pueda integrar en la aplicación desarrollada como un buscador más de la lista de motores agregados. Junto a esta posibilidad, aparece otra, que consiste en desarrollar diferentes modelos de recuperación de información sobre dicho índice, y realizar diversos experimentos de evaluación mediante la plataforma desarrollada con el fin de determinar aquellos algoritmos que devuelven mejores resultados.

Al ser muchas de las técnicas implementadas novedosas, es muy probable que en el futuro cercano se sigan descubriendo nuevos algoritmos, técnicas y métodos que permitan realizar las diferentes tareas de forma más efectiva. Por tanto, una posibilidad de ampliar el presente trabajo consiste en investigar nuevos algoritmos y técnicas para cada una de las funcionalidades implementadas, y realizar pruebas sobre el sistema para identificar aquellas técnicas o algoritmos que mejor funcionen sobre la aplicación.

Por último, dada la plataforma de evaluación desarrollada en el presente proyecto, es posible emplear la arquitectura desarrollada para probar las diferentes mejoras del sistemas, y comprobar así los beneficios que aportan a la aplicación. Un ejemplo de ello, sería realizar una comparación entre los ránkings diversificados y sin diversificar. Otra posibilidad sería realizar comparaciones entre sistemas que varíen los diferentes parámetros libres presentes en los algoritmos implementados, como los pesos de los buscadores en la obtención de scores para metabúsqueda.

6. Bibliografía

Agrawal, R., Gollapudi, S., Halverson, A., Ieong, S. Diversifying Search Results. 2nd Annual ACM International Conference on Web Search and Data Mining (WSDM 2009). Barcelona, February 2009, pp. 5-14.

Baeza-Yates, R., Ribeiro-Neto, B. Modern Information Retrieval - the concepts and technology behind search, 2nd Edition. Pearson Education Ltd., 2011.

Carbonell, J., Goldstein, J. The use of MMR, diversity-based reranking for reordering documents and producing summaries. 21st Annual ACM International SIGIR conference on Research and development in information retrieval (SIGIR 1998). Melbourne, August 1998, pp. 335-336.

Chapelle, O., Ji, S., Liao, E., Velipasaoglu, E., Lai, L., Wu, S.L. Intent-based Diversification of Web Search Results: Metrics and Algorithms. Information Retrieval, 14(6), December 2011, pp. 572-592.

Gayo-Avello, D., A survey on session detection methods in query logs and a proposal for future evaluation. Information Sciences 179(12), May 2009, pp. 1822-1843

Hagen, M., Stein B. and Rüb T. Query Session Detection as a Cascade. 20th Annual ACM International Conference on Information and Knowledge Management (CIKM 2011). Glasgow, October 2011, pp. 147-152.

He, D., Göker, A., Harper, D.J. Combining Evidence for Automatic Web Session Identification. Information Processing and Management 38(5), September 2002, pp. 727-742

Hoffman, K., Whiteson S. and de Rijke, M., A Probabilistic Method for Inferring Preferences from Clicks. 20th Annual ACM International Conference on Information and Knowledge Management (CIKM 2011). Glasgow, October 2011, pp. 249-258.

Hoffman, K., Whiteson, S. and de Rijke, M., Fidelity, Soundness and Efficiency of Interleaved Comparison Methods. Journal ACM Transactions on Information Systems (TOIS), 31(4), November 2013, Article no. 17.

Jansen, B.J., Spink, A., Blakely, C., Koshman, S. Defining a Session on Web Search Engines. Journal of the American Society for Information Science and Technology 58(6), April 2007.

Järvelin, K., Kekäläinen, J. Cumulated Gain-based Evaluation of IR Techniques. ACM Transactions on Information Systems (TOIS), 20(4), October 2002, pp. 422-446.

Kotov, A., Bennet, P.N., White, R.W., Dumais, S.T., Teevan, J. Modelling and Analysis of Cross-Session Search Tasks. 34th Annual International ACM SIGIR conference on Research and development in Information Retrieval (SIGIR 2011). Beijing, July 2011, pp. 5-14.

Lee, J. H. Analyses of Multiple Evidence Combination. 20th Annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 1997). Philadelphia, July 1997, pp. 267-276.

Markov, I., Arampatzis, A., Crestani, F. Unsupervised Linear Score Normalization Revisited. 35th Annual International ACM SIGIR conference on Research and development in information retrieval (SIGIR 2012). Portland, August 2012, pp. 1161-1162.

Montague, M., Aslam, J.A. Relevance Score Normalization for Metasearch. 10th Annual ACM International Conference on Information and Knowledge Management (CIKM 2001). Georgia, November 2001, pp. 427-433.

Rocchio, J.J. Relevance feedback in information retrieval. In Salton (1971), pp. 313-323.

Ruthven, I., Lalmas, M. A survey on the use of relevance feedback for information access systems. Knowledge Engineering Review, 18(2), June 2003, pp. 95-145.

Salton, G. 1971. The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall.

Salton, G., McGill, M.J. 1983. Introduction to Modern Information Retrieval. McGraw-Hill.

Spink, A., Jansen, B.J., Blakely, C., Koshman, S. A study of results overlap and uniqueness among major Web search engines. Information Processing and Management 42(5), September 2006, pp. 1379-1391.

Santos, R. L. T., MacDonald, C., Ounis, I. (2010 A) Selectively Diversifying Web Search Results. 19th Annual ACM International Conference on Information and Knowledge Management (CIKM 2010). Toronto, October 2010, pp. 1179-1188.

Santos, R. L. T., MacDonald, C., Ounis, I. (2010 B) Exploiting Query Reformulations for Web Search Result Diversification. 19th Annual International Conference on World Wide Web (WWW 2010). Raleigh, April 2010, pp. 881-890.

Schuth, A., Bruintjes, R., Büttner, F., van Doorn, J., Groenland, C, Tran., C., Oosterhuis, H., Veeling, V., van der Velde,J., Wechsler,R., Woudenberg, D., de Rijke, M. Probabilistic Multileave for Online Retrieval Evaluation. 38th Annual ACM International SIGIR conference on Research and development in information retrieval (SIGIR 2015). Santiago, August 2015.

Schuth, A., Sietsma, F., Whiteson, S., Lefortier, D., de Rijke, M., Multileaved Comparisons for Fast Online Evaluation. 23rd Annual ACM International Conference on Information and Knowledge Management (CIKM 2014). Shangai, November 2014, pp. 71-80.

Zhang Z., Sun, L., Han X., Learning to detect task boundaries of query session. 22nd Annual ACM International Conference on Information and Knoledge Management (CIKM 2013). San Francisco, October 2013, pp. 1885-1888.

Anexo 1: URLs y parámetros del servicio web

En este anexo, se detallan las URL y los parámetros del servicio web desarrollado en el presente trabajo.

Búsqueda

En este apartado, se detallan aquellas funciones del servicio web relativas a la búsqueda de documentos.

Metabúsqueda

URL: /search

Función: Realiza la búsqueda de resultados en los distintos sistemas de búsqueda agregados y construye el ranking unificado. Además, cambia la sesion interna si la nueva consulta no pertenece a la sesión activa.

Parámetros:

- query: Consulta a realizar
- clientIP: IP del usuario que va a realizar la consulta
- filter: Indica si se filtran o no los resultados guardados

Resultado: En caso de que todos los parámetros sean correctos y no se produzca ningún error, devuelve 3 listas de resultados. La primera, 'search', devuelve el ranking unificado resultado de la consulta. La segunda, 'clicked', consiste en la lista de resultados que el usuario ha seleccionado en la sesión de búsqueda actual, y 'saved' consta de los documentos guardados en la metasesión abierta por el usuario.

Los resultados de la lista 'search' contienen los siguientes campos:

- title: Título del documento
- url: URL en la que se encuentra el documento
- snippet: Breve descripción del documento
- rank: Posición del documento en el ranking unificado
- source: Lista de fuentes de las que procede el documento (identificadores de la tabla Ranker).
- value: Valor de la función de ranking
- saved: Indica si el documento se ha guardado previamente en la metasesión.

Los resultados de la segunda lista ('clicked') contienen los siguientes campos:

- title: Título del documento
- url: URL en la que se encuentra el documento
- snippet: Breve descripción del documento

Los resultados de la última lista ('saved') están formados por los siguientes campos:

- title: Título del documento
- url: URL en la que se encuentra el documento
- snippet: Breve descripción del documento
- query: Consulta de la que procede el documento
- rank: Rango del documento en el ranking del que procede

Además, se devuelve un campo llamado ‘new’ que toma el valor True si los documentos no se han recuperado de caché, y False en caso contrario.

En otro caso, devuelve un error 400 (Bad request) en caso de que los parámetros introducidos no sean correctos, o un error 500 (Internal Server Error) en caso de que se produzca algún error no esperado.

Registro de clicks

URL: /clicked

Función: Registra un documento como seleccionado por el usuario, y devuelve una lista con los documentos seleccionados en la sesión de búsqueda correspondiente.

Parámetros:

- query: Consulta de la que procede el documento seleccionado
- clientIP: IP del usuario que seleccionó el documento
- rank: Posición del documento en el ranking del que fue seleccionado

Resultado: Si no se produce ningún error, se devuelve una lista de resultados. Cada resultado consta de los siguientes campos:

- title: Título del documento
- url: URL en la que se encuentra el documento
- snippet: Breve descripción del documento

En el caso de que los parámetros no sean correctos, se devuelve un error 400 (Bad Request), o, si se produce un error inesperado, un error 500 (Internal Server Error).

Recuperación y filtro de resultados

URL: /filter/save

Función: Recupera los resultados de una búsqueda, filtrando aquellos resultados que el usuario ha guardado en una sesión explícita, si así se le indica.

Parámetros:

- query: Consulta a realizar
- clientIP: IP del usuario que va a realizar la consulta
- filter: Indica si se filtran o no los resultados guardados

Resultado: Si no se produce ningún error, devuelve el ranking unificado resultado de la consulta, quitando del mismo, si así se ha indicado, los resultados guardados por el usuario.

Los resultados de la lista contienen los siguientes campos:

- title: Título del documento
- url: URL en la que se encuentra el documento
- snippet: Breve descripción del documento
- rank: Posición del documento en el ranking unificado
- source: Lista de fuentes de las que procede el documento (identificadores de la tabla Ranker).
- value: Valor de la función de ranking
- saved: Indica si el documento se ha guardado previamente en la metasesión.

En caso de que los parámetros sean incorrectos, se devuelve un error 400 (Bad Request), o, si se produce un error inesperado, se devuelven error 500 (Internal Server Error).

Fuentes de los resultados

URL: /sources/

Función: Recupera las fuentes de los resultados de una búsqueda previa

Parámetros:

- query: Consulta a realizar
- clientIP: IP del usuario que va a realizar la consulta

Resultado: Devuelve una lista de fuentes de resultados, con los siguientes campos:

- rank: Posición del documento en el ranking de la búsqueda
- sources: Fuentes de las que procede el documento
- ranks: Posiciones del documento en los rankings previos a la unificación. En el mismo orden que el campo anterior.

En caso de que los parámetros sean incorrectos, se devuelve un error 400 (Bad Request), o, si se produce un error inesperado, se devuelven error 500 (Internal Server Error).

Relevance feedback

URL: /relevance/

Función: Añade una palabra a una consulta dada, y realiza la búsqueda correspondiente.

Parámetros:

- query: Consulta a realizar
- clientIP: IP del usuario que va a realizar la consulta
- filter: Indica si se filtran o no los resultados guardados

Resultado: En caso de que todos los parámetros sean correctos y no se produzca ningún error, devuelve 3 listas de resultados. La primera, 'linkList', devuelve el ranking unificado resultado de la consulta. La segunda, 'clicked', consiste en la lista de resultados que el usuario ha seleccionado en la sesión de búsqueda actual, y 'saved' consta de los documentos guardados en la metasesión abierta por el usuario.

Los resultados de la lista 'linkList' contienen los siguientes campos:

- title: Título del documento
- url: URL en la que se encuentra el documento
- snippet: Breve descripción del documento
- rank: Posición del documento en el ranking unificado
- source: Lista de fuentes de las que procede el documento (identificadores de la tabla Ranker).
- value: Valor de la función de ranking
- saved: Indica si el documento se ha guardado previamente en la metasesión.

Los resultados de la segunda lista ('clicked') contienen los siguientes campos:

- title: Título del documento
- url: URL en la que se encuentra el documento
- snippet: Breve descripción del documento

Los resultados de la última lista ('saved') están formados por los siguientes campos:

- title: Título del documento
- url: URL en la que se encuentra el documento
- snippet: Breve descripción del documento
- query: Consulta de la que procede el documento
- rank: Rango del documento en el ranking del que procede

Ademas, en el atributo 'query', se devuelve la consulta ampliada.

En caso de que los parámetros introducidos no sean correctos devuelve un error 400 (Bad request), y, en caso de que se produzca un error inesperado, devuelve un error 500 (Internal Server Error).

Diversidad

URL: /diversity/

Función: Calcula la matriz de similitudes entre los documentos recuperados para una determinada consulta

Parámetros:

- query: Consulta a realizar
- clientIP: IP del usuario que va a realizar la consulta

Resultado: En caso de que todos los parámetros sean correctos y no se produzca ningún error, devuelve la consulta realizada, en el campo 'query'. Además, en el campo 'matrix', se devuelve la matriz de similitudes, a partir de ternas, formadas por el ranking del primer documento ('rank1'), el ranking del segundo ('rank2') y la similitud entre ambos documentos ('distance'). Por motivos de eficiencia, solamente se devuelve la matriz diagonal inferior (la matriz es simétrica)

En otro caso, devuelve un error 400 (Bad request) en caso de que los parámetros introducidos no sean correctos, o un error 500 (Internal Server Error) en caso de que se produzca algún error no esperado.

Sesiones gestionadas por el usuario

En este apartado se detallan las llamadas al servicio web relativas a las sesiones explícitas o gestionadas por el usuario.

Guardar documento

URL: /metases/lnk/save

Función: Guarda un determinado documento en la metasesión correspondiente del usuario.

Parámetros:

- query: Consulta de la que procede el resultado
- clientIP: IP del usuario que va a realizar la consulta
- rank: Posición del documento en el ranking.

Resultado: Si el documento se guarda correctamente, devuelve la lista de los documentos guardados en la sesión. Cada documento en la lista devuelta tiene los siguientes campos:

- title: Título del documento
- url: URL en la que se encuentra el documento
- snippet: Breve descripción del documento
- query: Consulta de la que procede el documento
- rank: Rango del documento en el ranking del que procede

En otro caso, devuelve un error 400 (Bad request) en caso de que los parámetros introducidos no sean correctos, o un error 500 (Internal Server Error) en caso de que se produzca algún error no esperado.

Eliminar documento

URL: /metases/lnk/delete

Función: Elimina un determinado documento de la metasesión correspondiente del usuario.

Parámetros:

- query: Consulta de la que procede el resultado
- clientIP: IP del usuario que va a realizar la consulta
- rank: Posición del documento en el ranking.

Resultado: Si el documento se guarda correctamente, devuelve la lista de los documentos guardados en la sesión. Cada documento en la lista devuelta tiene los siguientes campos:

- title: Título del documento
- url: URL en la que se encuentra el documento
- snippet: Breve descripción del documento
- query: Consulta de la que procede el documento
- rank: Rango del documento en el ranking del que procede

En otro caso, devuelve un error 400 (Bad request) en caso de que los parámetros introducidos no sean correctos, o un error 500 (Internal Server Error) en caso de que se produzca algún error no esperado.

Cerrar metasesión

URL: /metases/close

Función: Cierra la metasesión actual del usuario

Parámetros:

- clientIP: IP del usuario que controla la metasesión a cerrar.

Resultado: Devuelve un error 400 (Bad request) en caso de que los parámetros introducidos no sean correctos, o un error 500 (Internal Server Error) en caso de que se produzca algún error no esperado.

Si se cierra correctamente la sesión, devuelve un JSON vacío.

Abrir metasesión

URL: /metases/open

Función: Abre una sesión anterior, y devuelve los resultados guardados en la misma.

Parámetros:

- clientIP: IP del usuario que controla la metasesión a abrir.
- metaName: Nombre de la sesión

Resultado:

En caso de que la sesión exista, devuelve la lista de resultados guardados. Cada resultado guardado tiene los siguientes campos:

- title: Título del documento
- url: URL en la que se encuentra el documento
- snippet: Breve descripción del documento
- query: Consulta de la que procede el documento
- rank: Rango del documento en el ranking del que procede

En caso de que la metasesión correspondiente no exista, retorna un error 404 (Not found). Devuelve un error 400 (Bad request) en caso de que los parámetros introducidos no sean correctos, o un error 500 (Internal Server Error) en caso de que se produzca algún error no esperado.

Comprobar nombre

URL: /metases/available

Función: Comprueba si el usuario ha guardado previamente una sesión con el nombre indicado.

Parámetros:

- clientIP: IP del usuario.
- metaName: Nombre de la sesión

Resultado: Devuelve, en un campo 'available', el valor True si no existe ninguna sesión con el nombre indicado, o la sesión con dicho nombre es la sesión actual, o el valor False si existe una sesión distinta.

Devuelve un error 400 (Bad request) en caso de que los parámetros introducidos no sean correctos, o un error 500 (Internal Server Error) en caso de que se produzca algún error no esperado.

Guardar sesión

URL: /metases/save

Función: Proporciona un nombre a la sesión para que pueda ser recuperada posteriormente, y la cierra.

Parámetros:

- clientIP: IP del usuario que desea guardar la sesión.
- metaName: Nombre nuevo de la sesión.

Resultado: Devuelve un JSON vacío si todo sale bien.

Devuelve un error 400 (Bad request) en caso de que los parámetros introducidos no sean correctos, o un error 500 (Internal Server Error) en caso de que se produzca algún error no esperado.

Evaluación

En este apartado, se muestran las URLs y métodos para la evaluación de los diferentes sistemas.

Búsqueda e intercalado de rankings

URL: evaluation/search

Función: Realiza la búsqueda de resultados en los distintos sistemas de búsqueda agregados a evaluar y construye el ranking unificado, intercalando los resultados.

Parámetros:

- query: Consulta a realizar
- clientIP: IP del usuario que va a realizar la consulta

Resultado: Devuelve una lista con el ranking intercalado ('search'). Los resultados de la lista 'search' contienen los siguientes campos:

- title: Título del documento
- url: URL en la que se encuentra el documento
- snippet: Breve descripción del documento
- rank: Posición del documento en el ranking unificado
- source: Lista de fuentes de las que procede el documento (identificadores de la tabla Ranker).
- value: Valor de la función de ranking

Registro de clicks

URL: evaluation/clicked

Función: Registra un documento como seleccionado por el usuario, lo marca para evaluación, y devuelve una lista con los documentos seleccionados en la sesión de búsqueda correspondiente.

Parámetros:

- query: Consulta de la que procede el documento seleccionado
- clientIP: IP del usuario que seleccionó el documento
- rank: Posición del documento en el ranking del que fue seleccionado

Resultado: Si no se produce ningún error, se devuelve una lista de resultados. Cada resultado consta de los siguientes campos:

- title: Título del documento
- url: URL en la que se encuentra el documento
- snippet: Breve descripción del documento

En el caso de que los parámetros no sean correctos, se devuelve un error 400 (Bad Request), o, si se produce un error inesperado, un error 500 (Internal Server Error).

Evaluación por clicks

URL: evaluation/evaluate/clicks

Función: Realiza la evaluación de los diferentes resultados por medio del número esperado de clicks para cada sistema, a partir de los documentos seleccionados por los usuarios en el sistema de evaluación.

Parámetros: No necesita ningún parámetro.

Resultado: Devuelve una lista de buscadores, con los siguientes campos:

- name: Nombre del buscador
- evaluation: Evaluación del buscador

Evaluación por consultas

URL: evaluation/evaluate/queries

Función: Realiza la evaluación de los diferentes resultados por medio del número esperado de consultas ganadas por cada sistema, a partir de los documentos seleccionados por los usuarios en el sistema de evaluación.

Parámetros: No necesita ningún parámetro.

Resultado: Devuelve una lista de buscadores, con los siguientes campos:

- name: Nombre del buscador
- evaluation: Evaluación del buscador

Evaluación por P@k

URL: evaluation/evaluate/patk

Función: Realiza la evaluación de los diferentes resultados por medio de la métrica *precission at k*, a partir de los documentos seleccionados por los usuarios en el sistema de evaluación.

Parámetros:

- k: Punto de corte de cada ranking.

Resultado: Devuelve una lista de buscadores, con los siguientes campos:

- name: Nombre del buscador
- evaluation: Evaluación del buscador

En el caso de que el parámetro sea incorrecto, se devuelve un error 400 (Bad Request).

Evaluación por R@k

URL: evaluation/evaluate/ratk

Función: Realiza la evaluación de los diferentes resultados por medio de la métrica *recall at k*, a partir de los documentos seleccionados por los usuarios en el sistema de evaluación.

Parámetros:

- k: Punto de corte de cada ranking.

Resultado: Devuelve una lista de buscadores, con los siguientes campos:

- name: Nombre del buscador
- evaluation: Evaluación del buscador

En el caso de que el parámetro sea incorrecto, se devuelve un error 400 (Bad Request).

Evaluación por nDCG@k

URL: evaluation/evaluate/ndcg

Función: Realiza la evaluación de los diferentes resultados por medio de la métrica *negative discounted cumulative gain at k*, a partir de los documentos seleccionados por los usuarios en el sistema de evaluación.

Parámetros:

- k: Punto de corte de cada ranking.

Resultado: Devuelve una lista de buscadores, con los siguientes campos:

- name: Nombre del buscador
- evaluation: Evaluación del buscador

En el caso de que el parámetro sea incorrecto, se devuelve un error 400 (Bad Request).

Evaluación por MRR

URL: evaluation/evaluate/patk

Función: Realiza la evaluación de los diferentes resultados por medio de la métrica *mean reciprocal rank*, a partir de los documentos seleccionados por los usuarios en el sistema de evaluación.

Parámetros: No necesita ningún parámetro.

Resultado: Devuelve una lista de buscadores, con los siguientes campos:

- name: Nombre del buscador
- evaluation: Evaluación del buscador

Evaluación por MAP

URL: evaluation/evaluate/map

Función: Realiza la evaluación de los diferentes resultados por medio de la métrica *mean average precission*, a partir de los documentos seleccionados por los usuarios en el sistema de evaluación.

Parámetros: No necesita ningún parámetro.

Resultado: Devuelve una lista de buscadores, con los siguientes campos:

- name: Nombre del buscador
- evaluation: Evaluación del buscador

Evaluación por ILD

URL: evaluation/evaluate/ild

Función: Realiza la evaluación de los diferentes resultados por medio de la métrica *intra-list dissimilarity at 10*, a partir de los documentos seleccionados por los usuarios en el sistema de evaluación.

Parámetros: No necesita ningún parámetro.

Resultado: Devuelve una lista de buscadores, con los siguientes campos:

- name: Nombre del buscador
- evaluation: Evaluación del buscador

Informe de evaluación

URL: evaluation/evaluate/report

Función: Realiza el cálculo de todas las métricas para el sistema. Concretamente, calcula el número esperado de clicks, el número de consultas ganadas por cada uno de los sistemas, P@5, P@10, R@5, R@10, nDCG@5, nDCG@10, MAP, MRR e ILD@10.

Parámetros: No necesita ningún parámetro.

Resultado: Devuelve diversos campos, cada uno representando una de las métricas calculadas. Cada una de las métricas está representada por un diccionario, que tiene como claves los nombres de los buscadores, y como valores, los valores de la métrica correspondiente para dicho buscador. Los campos devueltos son los siguientes:

- clicks: Número esperado de clicks para cada uno de los sistemas.
- queries: Número de consultas ganadas por cada uno de los sistemas.
- PAt5: Valores de la métrica Precission@5.
- PAt10: Valores de la métrica Precission@10
- RAt5: Valores de la métrica Recall@5.
- RAt10: Valores de la métrica Recall@10.
- nDCGAt5: Valores de la métrica nDCG@5.
- nDCGAt10: Valores de la métrica nDCG@10.
- MAP: Valores de la métrica MAP.
- MRR: Valores de la métrica MRR.
- ILD: Valores de la métrica ILD@10.

Anexo 2: Diagramas UML

En este anexo se muestran los diferentes diagramas de clases para los distintos módulos de la aplicación.

Servicio web

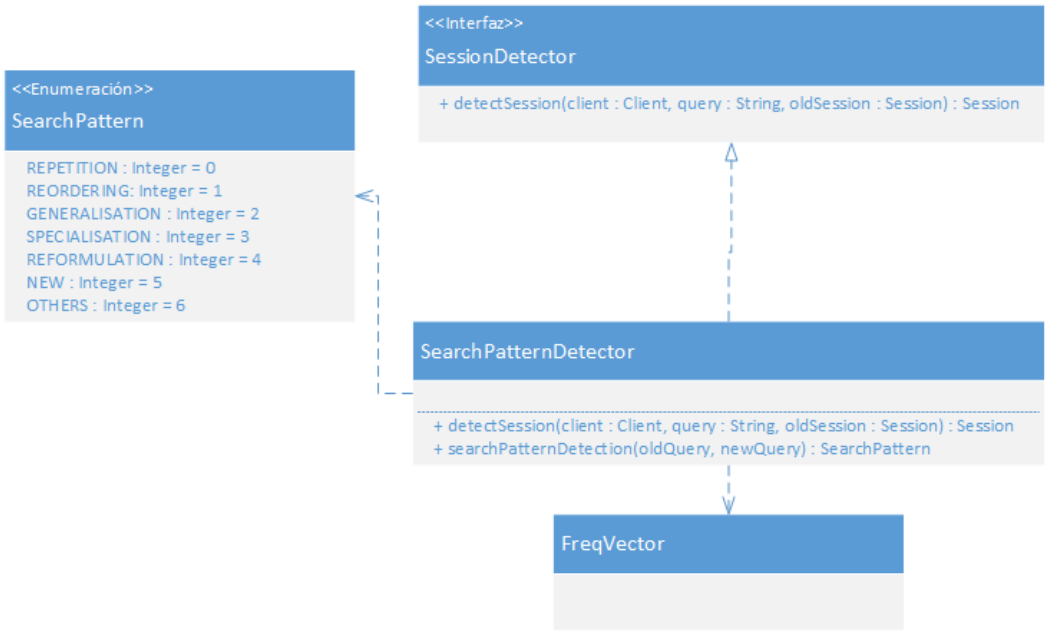

Figura 22. Diagrama de clases del módulo Session Detection

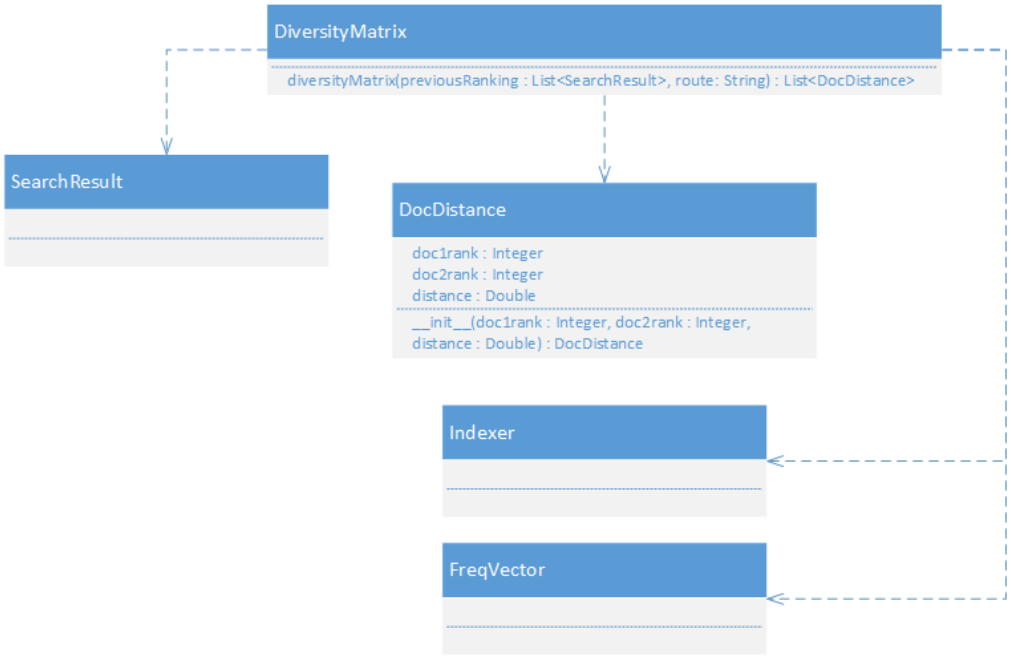

Figura 23. Diagrama de clases del módulo Diversity

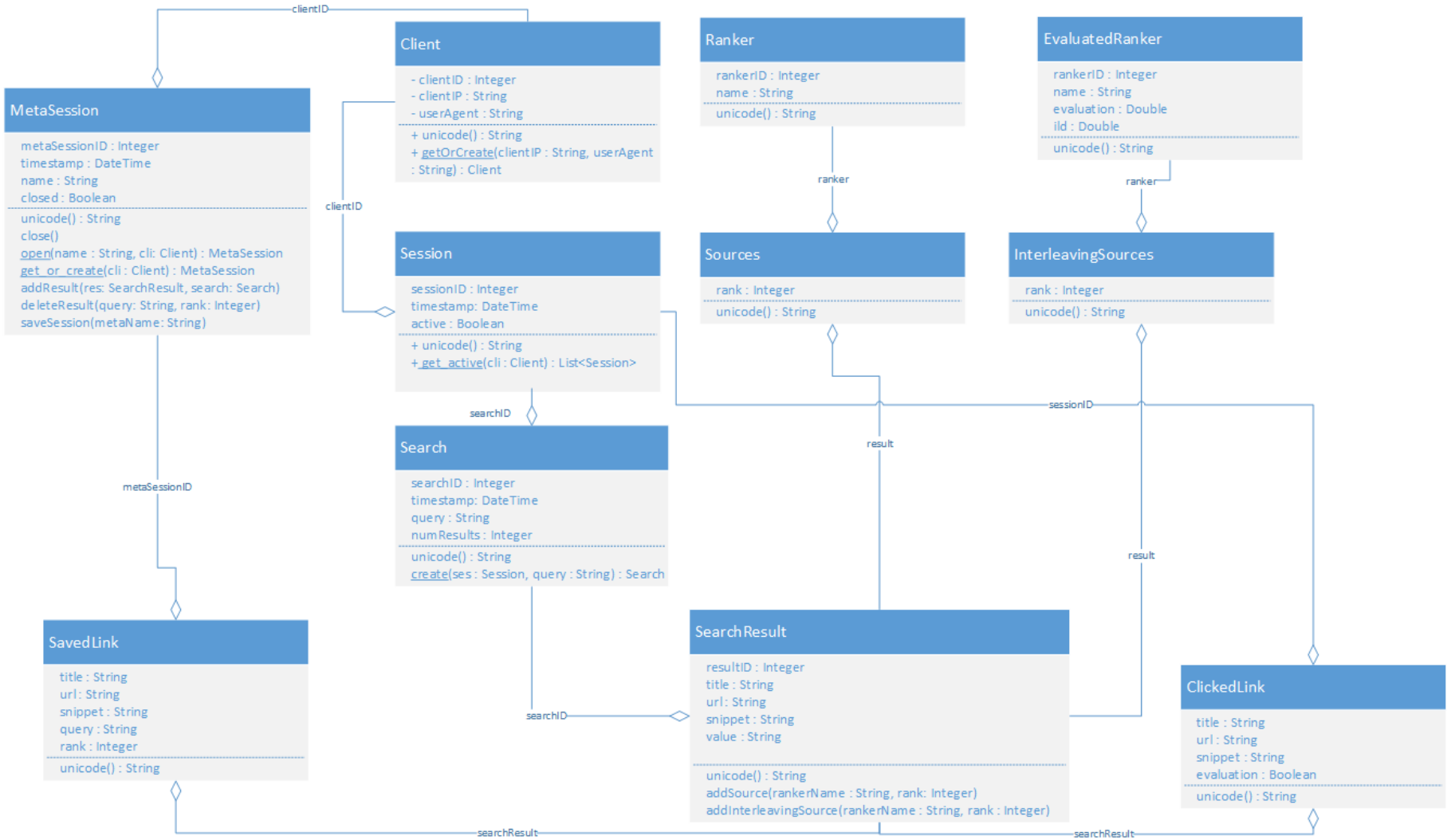

Figura 24. Diagrama de clases de los modelos

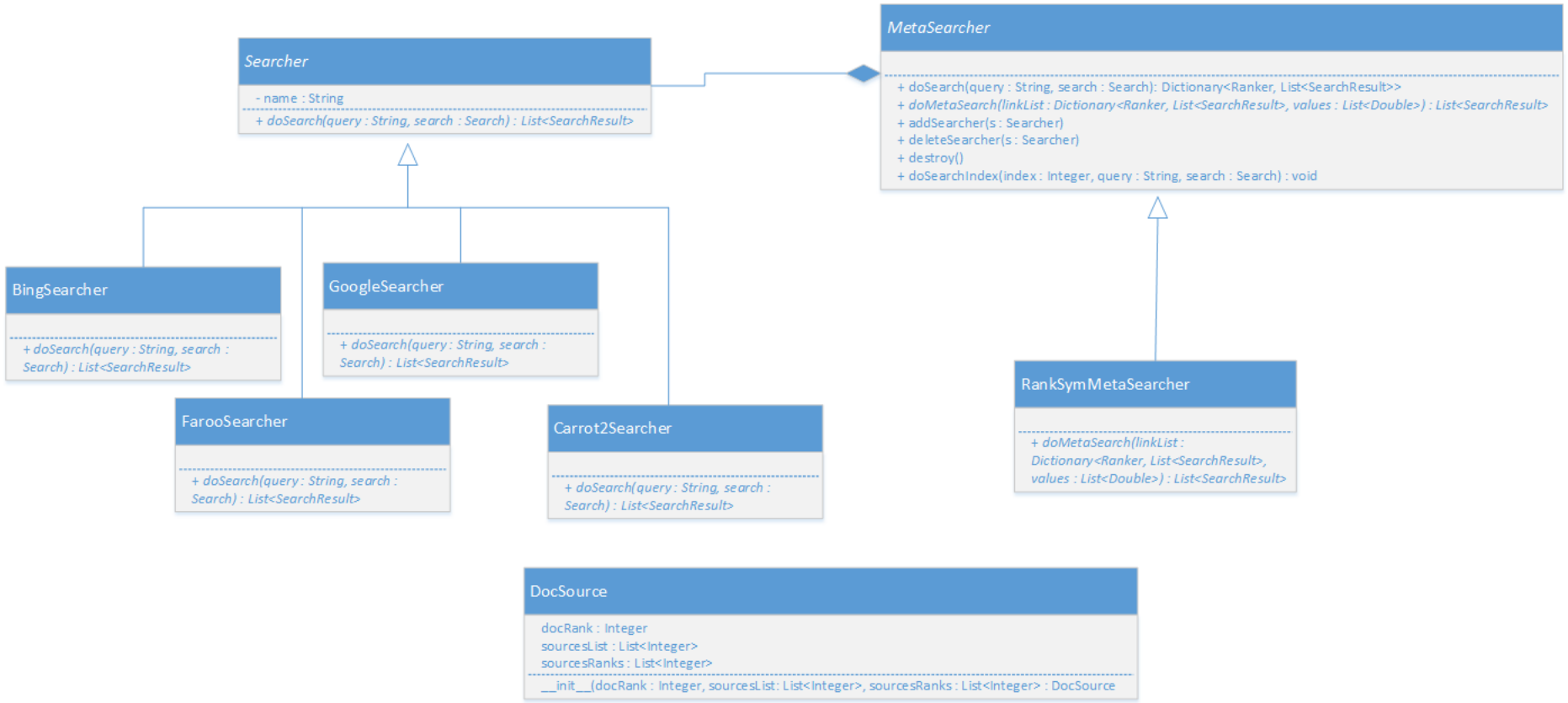

Figura 25. Diagrama de clases del módulo MetaSearch del servicio web

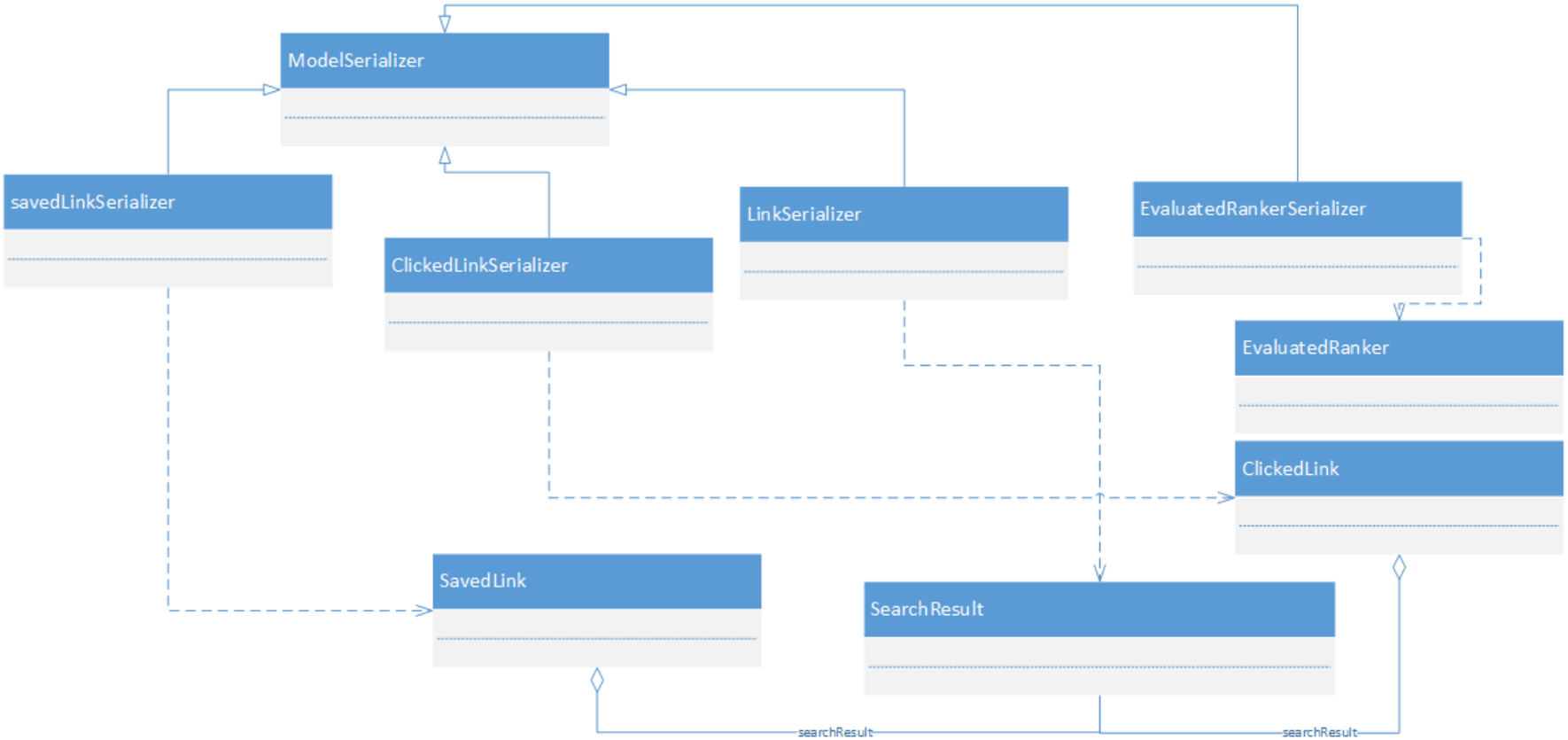

Figura 26. Diagrama de clases de los serializadores del servicio web

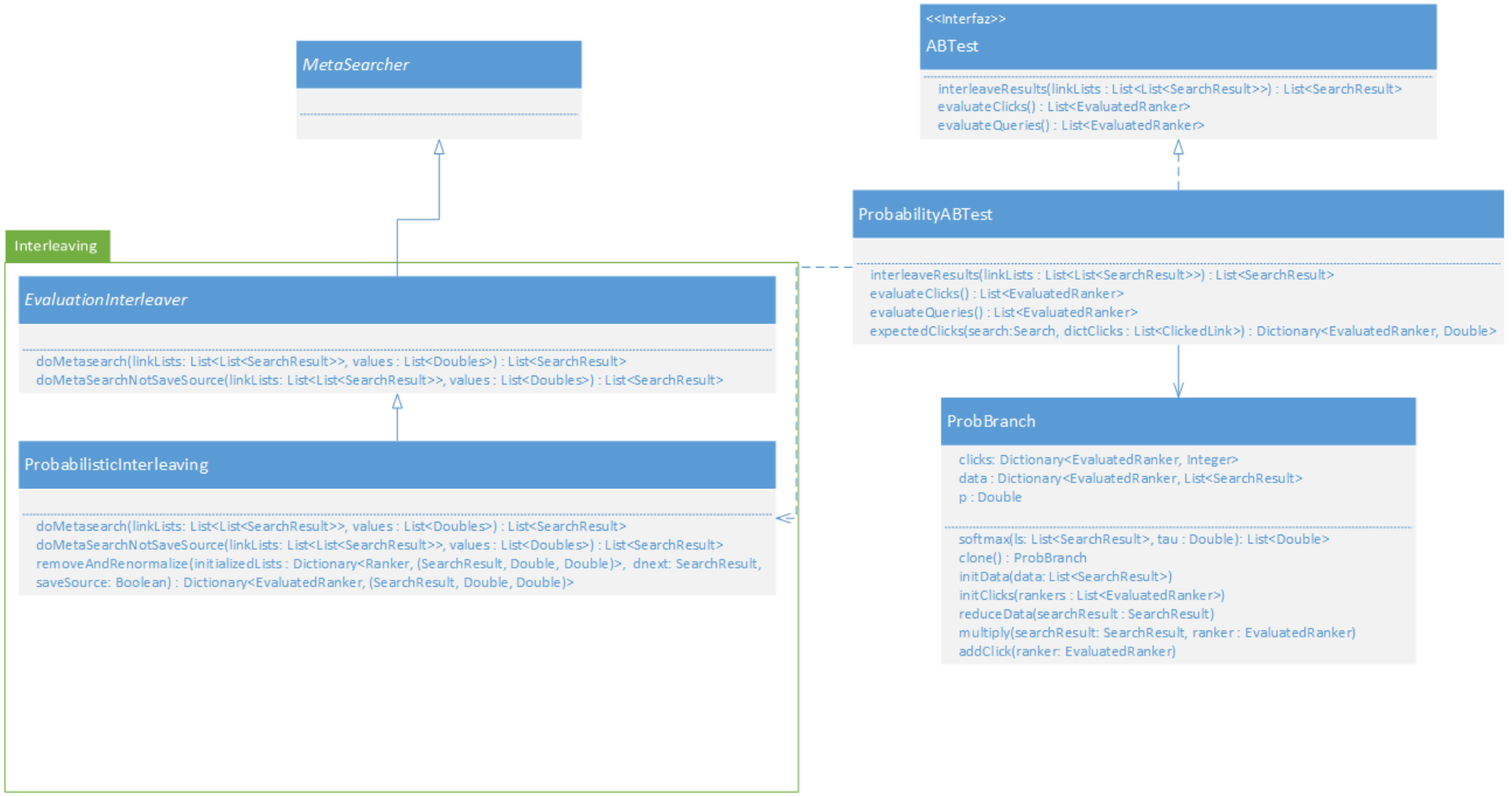

Figura 27. Diagrama de clases del módulo de evaluación (sin métricas)

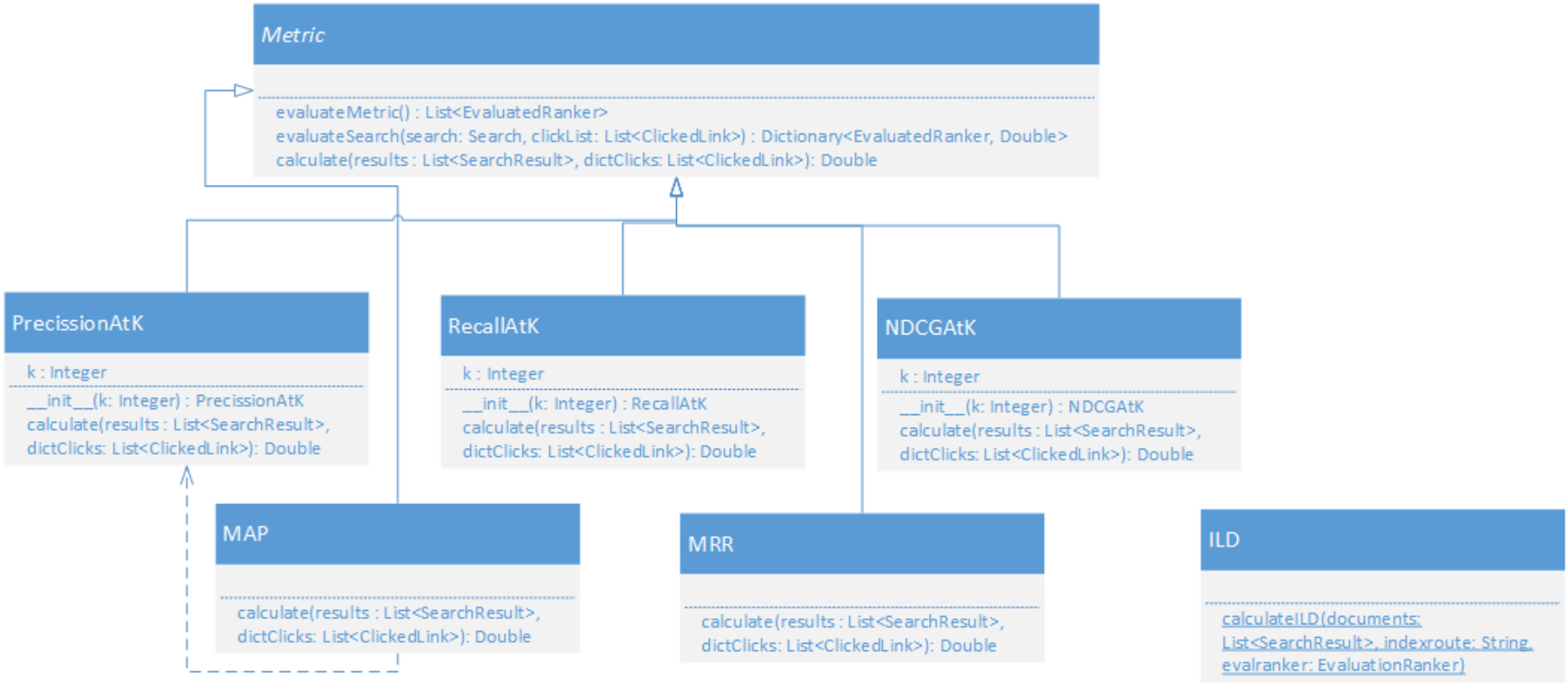

Figura 28. Diagrama de clases del submódulo Metrics del módulo Evaluation

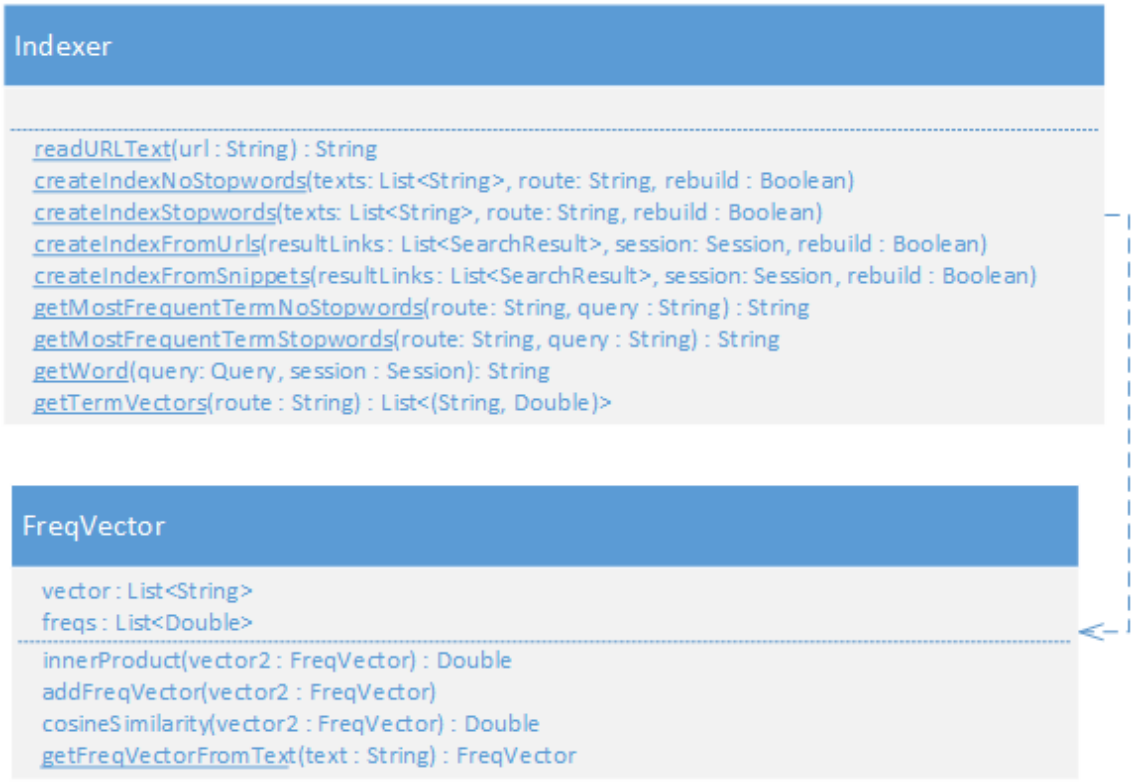

Figura 29. Diagrama de clases del módulo Indexing.

Cliente web

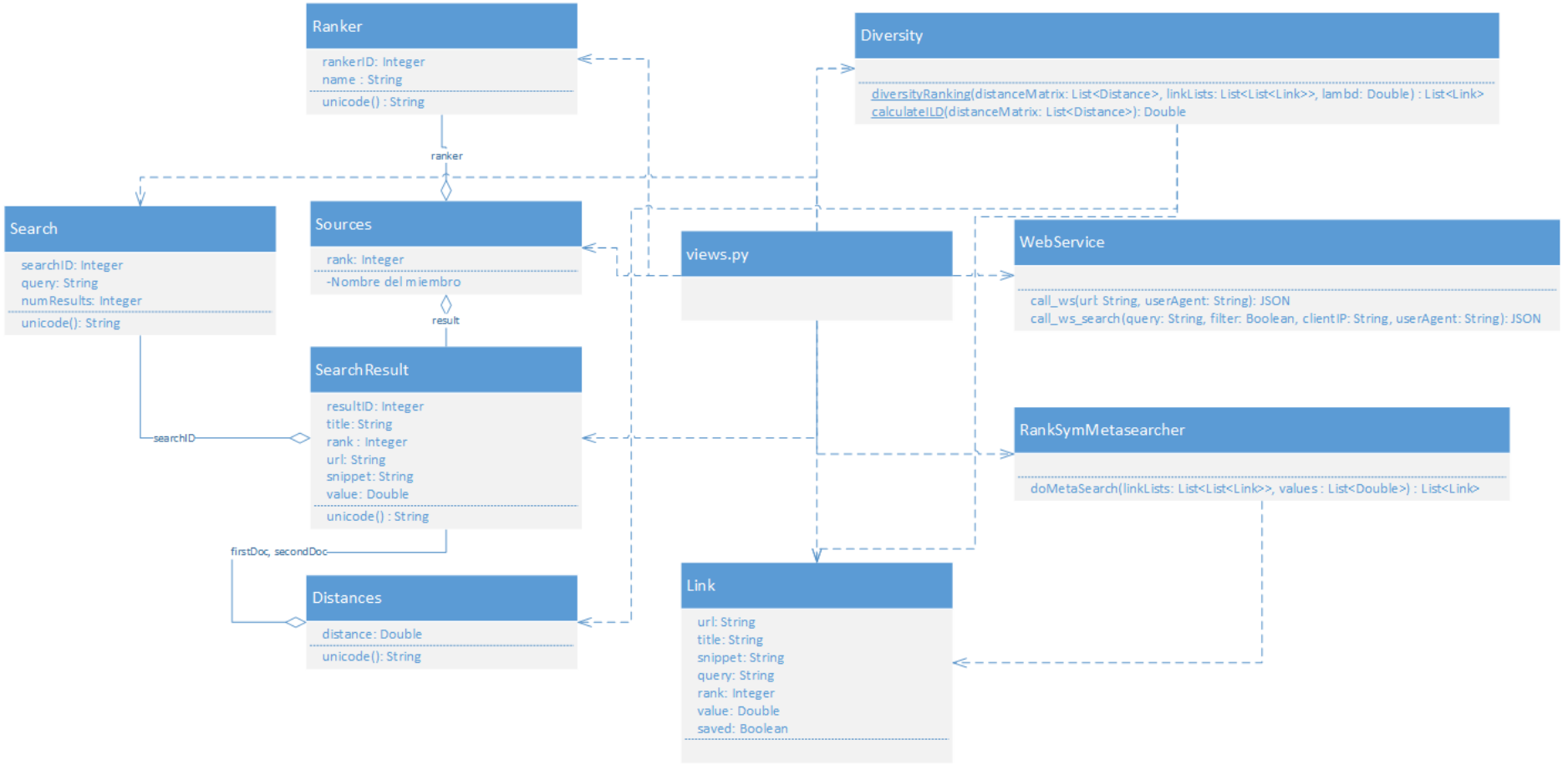

Figura 30. Diagrama de clases del cliente web.