UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TAG CLOUDS PARA INVESTIGADORES DE CIENCIAS DE LA COMPUTACIÓN

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERA CIVIL EN COMPUTACIÓN

PAULA ANDREA RÍOS ARAYA

PROFESOR GUÍA:
AIDAN HOGAN

MIEMBROS DE LA COMISIÓN:
BÁRBARA POBLETE LABRA
MARCELA CALDERÓN CORAIL

SANTIAGO DE CHILE
2018

Resumen

Actualmente, existen millones de publicaciones de investigadores en distintas áreas de las Ciencias de la Computación, y estas continúan aumentando día a día. En los perfiles de cada investigador del área en sitios web como DBLP o Google Scholar, se puede encontrar un listado con sus publicaciones. Sin embargo, con esta información por sí sola es difícil captar cuáles son los tópicos de interés de cada investigador a simple vista, y podría ser necesario en un ámbito de colaboración entre académicos o entre académicos y estudiantes.

Este trabajo busca facilitar la información resumida de los tópicos de investigación de académicos de Ciencias de la Computación mediante la generación de visualizaciones como nubes de palabras, o *tag clouds*, a partir de las palabras y frases clave mencionadas en las publicaciones encontradas en repositorios bibliográficos online, como los mencionados anteriormente.

El sistema desarrollado en esta memoria consiste en una herramienta que permite la creación de *tag clouds* para perfiles de DBLP. Esta herramienta se encarga de la obtención de las publicaciones encontradas en el perfil, la extracción de potenciales *keywords* y la selección de las *keywords* más relevantes según cuatro modelos de ordenamiento. Por cada uno de estos modelos se crea una variante de *tag cloud*. Además, se crea un sitio web que permite el uso de la herramienta para cualquier usuario.

El trabajo se enfoca principalmente en la investigación de modelos de *learning to rank* y la comparación de su desempeño en la tarea de definir las *keywords* más relevantes para un investigador de Ciencias de la Computación. Dado que existen tres enfoques distintos para resolver la tarea de ordenamiento, se utilizan cuatro modelos de *learning to rank*, teniendo al menos uno por cada enfoque. Estos son regresión lineal, RankSVM, LambdaMART y AdaRank.

De las evaluaciones a las *tag clouds* creadas por la herramienta se observa que no habría una preferencia absoluta por un método por sobre los demás, sino que varía según cada persona, pero en la mayoría de los casos se le asigna el puntaje máximo a al menos una de las *tag clouds* generadas. Esto podría deberse a que los modelos tienden a diferir en su enfoque, en algunos casos seleccionando *keywords* más técnicas y en otros más genéricas. De esta forma la apreciación de un método por sobre el otro se ve afectada por las preferencias de cada uno. De esto se concluye la importancia de dar la posibilidad de elegir a los usuarios entre distintas variantes.

A la Kimi.

Agradecimientos

A mi familia por el apoyo de todos estos años, en particular a mis padres por nunca desalentarme en mi interés en la ciencia y siempre creer en mis habilidades. Es gracias a esto que me encuentro acá y me siento privilegiada, especialmente porque sé que no es la realidad de muchas niñas, incluso hoy en día.

A mi profesor guía Aidan Hogan por apoyarme y orientarme en este trabajo de memoria, resolviendo todas mis dudas.

Al Coro de la FCFM y el Coro Sinfónico de la Universidad de Chile y todos sus miembros, por ser las mejores distracciones y enseñarme tanto. En especial, el coro de la facultad y su directora, Verónica Rivas, fueron una parte esencial de mis años en esta universidad y tendrán un lugar en mi corazón siempre.

A *queue*[/'kwɛ̃u.e/] por todos estos años de "Club de Cocina", gorritos de cumpleaños (que a veces no eran gorritos) y almuerzos en física.

A Tomás Díaz por ser mi compañero y apoyo constante. Y por muchas, muchas cosas más. No podría ponerlas todas acá.

Y a Dios por todo lo demás.

Tabla de contenido

Índice de tablas

Índice de ilustraciones

Capítulo 1

Introducción

Actualmente, existen millones de publicaciones de investigadores en distintas áreas de las Ciencias de la Computación, y estas continúan aumentando día a día. En particular, DBLP, sitio web que posee un enorme repositorio bibliográfico de artículos relacionados con Ciencias de la Computación, cuenta con más de 4,2 millones de publicaciones de más de 2 millones de publicadores[1].

En los perfiles de cada investigador en sitios web como DBLP, o en otros más generales como Google Scholar, se puede encontrar un listado con sus publicaciones, las cuales se pueden filtrar según año de publicación, tipo (libros, tesis, artículos, etc.), co-autor, entre otros. Sin embargo, con esta información por sí sola es difícil captar cuáles son los tópicos de interés de cada investigador a simple vista, y considerando que en varios casos cuentan con un gran número de publicaciones y la gran cantidad de temas y áreas que cubren las Ciencias de la Computación, esto se vuelve aún más complicado.

Lo que se busca con este trabajo es facilitar la información resumida de los tópicos de investigación de académicos de Ciencias de la Computación mediante la generación de visualizaciones como nubes de palabras, o *tag clouds*, a partir de las *keywords*[2] en las publicaciones encontradas en repositorios bibliográficos online como los mencionados anteriormente.

1.1. Motivación

En el contexto de la academia, la colaboración entre académicos para la realización de una investigación y posterior publicación es algo bastante común y que puede tener grandes beneficios para ambas partes. Por otra parte, también es común la colaboración entre académicos y alumnos como, por ejemplo, en el caso de un trabajo de título, para lo cual los estudiantes buscan a un profesor guía que los pueda orientar, o trabajos dirigidos, en los

[1] https://dblp.uni-trier.de/

[2] Algunos autores distinguen las nociones de *keywords* (palabras claves) y *key-phrases* (frases clave), considerando la primera como compuesta por una palabra y la segunda como una o más palabras. En este caso, se les referirá en ambos casos como *keywords*.

Figura 1.1: Ejemplo de nube de palabras generado con WordClouds.com

cuales los alumnos buscan trabajar durante un periodo junto con un profesor. Es en estos ámbitos, entre otros, que podría ser de gran interés tener acceso a una herramienta que permita captar rápidamente una idea de cuáles son los principales temas de interés o el área de especialización de posibles colaboradores.

Las visualizaciones de datos (como gráficos, infografías, entre otros) permiten comunicar información importante con un vistazo. En el caso de datos basados en texto, una forma efectiva de resaltar puntos importantes es mediante el uso de *tag clouds*: formato de visualización de datos textuales que consiste en la representación del conjunto de palabras más frecuentes de un texto, con un pre-procesamiento previo en el cual se extraen las palabras sin contenido del texto, donde la importancia de cada una se refleja en el tamaño de la fuente. Este formato es bastante útil para percibir rápidamente los términos más relevantes, o comparar la relevancia relativa de distintos términos.

A pesar de que ya existen muchas herramientas disponibles en línea, como WordClouds.com[3], que permiten crear *tag clouds* a partir de textos, o incluso directamente de archivos PDF, que es el formato en el que suelen encontrarse los papers académicos en repositorios online, estas en general dividen frases (como, por ejemplo, "Semantic Web") en palabras individuales, con lo cual se pierde el sentido de la frase y no es muy útil para el propósito planteado. Otro tema a considerar, es que se requiere, además, que la herramienta acepte múltiples archivos, y no solo uno, para obtener las *keywords*.

Como se puede observar en el ejemplo (ver Figura 1), esta es una nube de palabras generada a partir del texto encontrado en un paper sobre IMGpedia y el enriquecimiento de la Web de Datos con análisis de contenido de imagen[4]. Aunque la imagen permite captar una idea general de qué trata el texto, muchas de las frases clave se ven separadas (como "Web of Data", "Linked Data", "neural network", entre otras) y no se filtran palabras del lenguaje común (como "using", "different", etc.), aunque estas pueden ser manualmente removidas.

En el contexto de colaboración entre académicos, existen plataformas para este fin como

[3]https://www.wordclouds.com/
[4]http://dblp.org/rec/conf/amw/FerradaBH16

ResearchGate o Academia.edu, las cuales están planteadas más bien como redes sociales para académicos y requieren una cuenta para poder ingresar. Sin embargo, estas no cuentan con visualizaciones que permitan captar a simple vista el contenido de las publicaciones de un autor, para lo cual podría llegar a ser útil la presencia de una *tag cloud*.

La solución que se plantea consiste en crear una herramienta propia que permita la creación de *tag clouds* para perfiles de DBLP. Para esto se quiere capturar las publicaciones de un perfil y descargarlas, pasarlas a texto, realizar una limpieza y pre-procesamiento para luego extraer *keywords*, seleccionando las más relevantes. Finalmente, se quiere generar visualizaciones de interés, en particular *tag clouds*, con estas, para lo cual además se requiere generar un ordenamiento para las *keywords* según su relevancia, el cual define el tamaño de la letra utilizada para cada una. Este proceso plantea varios objetivos, desde bajar las publicaciones, las cuales se encuentran en diversos sitios web distintos, hasta la definición de un ordenamiento para las *keywords*. Para esto último se requiere investigar posibles técnicas de ordenamiento que se puedan aplicar a la tarea de definir cuáles son las *keywords* más relevantes para un investigador de Ciencias de la Computación.

1.2. Objetivos de la memoria

El objetivo general de este trabajo de memoria es el desarrollo de una herramienta accesible como servicio en línea que permita crear visualizaciones de *keywords* a partir de perfiles de investigadores de Ciencias de la Computación en repositorios bibliográficos en línea.

Por lo tanto, los objetivos específicos que se desprenden del objetivo general antes enunciado, son los siguientes:

- Obtener las publicaciones encontradas en perfiles de investigadores de Ciencias de la Computación en DBLP en formato PDF.
- Realizar un *scraping*[5] de las páginas de las editoriales en las cuales se encuentran las publicaciones.
- Realizar una limpieza y procesamiento del texto previo.
- Extraer *keywords* a partir del texto mediante el uso de distintas métricas.
- Crear un "*gold standard*" o un conjunto ideal de *keywords* para un grupo de autores.
- Utilizar técnicas de aprendizaje automático, en particular modelos de ordenamiento, para determinar las palabras más relevantes y definir un orden para estas según su relevancia.
- Generar visualizaciones y probar con distintas variantes.
- Validar la relevancia de las visualizaciones generadas con académicos del Departamento de Ciencias de la Computación (DCC) de la universidad.
- Crear sitio web que permita el uso de la herramienta para cualquier usuario.

[5] *Web scraping* es el proceso de extracción de información de sitios web. En particular en este trabajo se busca realizar este proceso de forma automática.

Capítulo 2

Marco teórico

En este capítulo se introducen los conceptos involucrados en el trabajo de memoria realizado. En primer lugar, se describe en qué consisten las *tag clouds* y los distintos aspectos a considerar en su uso. Luego de esto, se muestran técnicas de extracción de *keywords*, necesarias para poblar las *tag clouds*. Por otra parte, se describen conceptos generales de aprendizaje automático, los cuales son aplicados en las técnicas enfocadas en resolver tareas de ordenamiento, referidas como *learning to rank*. Estas técnicas son las utilizadas para definir un orden de relevancia entre las *keywords* extraídas. Finalmente, se muestran los tres principales enfoques de *learning to rank* y métricas para la evaluación de su desempeño.

2.1. *Tag Clouds*

Las nubes de etiquetas o *tag clouds* son un formato de visualización de datos textuales que consiste en la representación del conjunto de palabras más importantes de un texto. A diferencia de otras visualizaciones, las nubes de etiquetas no usan artefactos adicionales para representar variables de interés, sino que se representa con la palabra misma, mediante manipulaciones a las propiedades visuales del texto [1].

Existen varias características del texto que se pueden modificar para representar su relevancia, como el color del texto, el tamaño del texto, el orden de las palabras o la posición de estas. En particular, el tamaño del texto es el enfoque más común utilizado para este fin, ya que en general tiene un mayor impacto en cómo se procesa visualmente la información de una nube de palabras, siendo las palabras con tamaño más grande las más fáciles de captar y retener [1].

Se han realizado varios estudios sobre la efectividad del uso de *tag clouds* como fuente de información, en particular un estudio en el cual se analiza la utilidad de nubes de etiquetas para el resumen de resultados de búsqueda de consultas sobre un texto biomédico. Se mostró que "la interfaz de nube de etiquetas es ventajosa en la presentación de información descriptiva y en la reducción de la frustración del usuario" [9]. Sin embargo, existen varias críticas contra

las *tag clouds*[1], pero estas principalmente se fundan en que las *tag clouds* en varias ocasiones se ven contaminadas por el uso de palabras del lenguaje común como *tags*, en lugar de palabras o frases "clave" que revelen el contenido del texto, que es lo que se pretende usar.

Aun así, es necesario tener en consideración el contexto en el que son utilizadas, ya que para ciertos objetivos existen formas mejores de presentar esta información. Por ejemplo, si el objetivo es encontrar una palabra definida en un conjunto, una lista ordenada sería una mejor opción [6]. Por otra parte, cuando se trata de visualizar rápidamente algunos patrones generales en el texto [13] o mostrar términos relevantes de un texto de una forma estéticamente agradable, las *tag clouds* pueden ser una buena opción, y suelen ser bien recibidas. Esto podría ser debido a que en general los usuarios escanean visualmente las *tag clouds* de forma rápida, en vez de leer cada término [12], por lo cual las palabras cuyo texto es más grande captan la atención de los usuarios con efectividad.

Tag clouds para investigadores

Recientemente, el sitio web de la Facultad de Ciencias Físicas y Matemáticas[2] comenzó a incluir *tag clouds* en las páginas de los académicos. Aunque no se sabe con certeza cuál es la metodología utilizada para su creación, se cree que estas están basadas en los títulos de las publicaciones disponibles, lo cual difiere del enfoque planteado en este trabajo. Por otra parte, cabe mencionar que no todos los profesores cuentan con una, aunque probablemente serán añadidas a futuro. Dado a las metodologías distintas utilizadas, se cree que la herramienta creada para este trabajo de memoria podría ser utilizada para el perfil de los académicos de forma complementaria.

2.2. Técnicas de extracción de *keywords*

Keywords y *key-phrases*, o palabras y frases clave, son los términos utilizados para definir las frases o palabras que representan la información más relevante contenida en un documento y permiten caracterizar los tópicos discutidos en el documento. De esta forma, estas palabras permiten captar una idea de qué trata un texto sin necesidad de leerlo.

Normalmente, las *keywords* son escogidas de forma manual por los autores del documento, lo cual puede ser una tarea tediosa, en particular si se quiere hacer para varios documentos. Es por esto que en años recientes se han explorado, y se siguen explorando, formas de realizar esta tarea de forma automatizada. Existen varias metodologías propuestas para la extracción de *keywords*, de los cuales se realizan experimentos con tres de ellas, descritas a continuación.

[1] http://www.niemanlab.org/2011/10/word-clouds-considered-harmful/
[2] http://ingenieria.uchile.cl/

2.2.1. Tf-idf

Term Frequency - Inverse Document Frequency, o frecuencia de término - frecuencia inversa de documento, es una métrica utilizada para determinar la importancia de términos en un documento basado en la frecuencia con la que aparecen dentro de un corpus[3]. Esto se basa en la idea de que si un término aparece con frecuencia es importante, sin embargo, si aparece en muchos documentos no es un identificador único, por lo que no sería relevante en el conjunto. Tf-idf se utiliza comúnmente como una herramienta para extraer las *keywords* de un documento, no solo considerando los n-gramas[4] de dicho documento, sino de todos los documentos del corpus.

La frecuencia de término tf(t, d) se puede definir simplemente como la frecuencia bruta del término, es decir, la cantidad de veces que un término t ocurre en un documento d. También se puede definir como una "frecuencia normalizada", para evitar una predisposición hacia documentos largos, por ejemplo, una relación de las ocurrencias de un término en un documento y el número de ocurrencias de la palabra más frecuente dentro del mismo documento.

Por otra parte, la frecuencia inversa de documento idf(t, D) se utiliza para medir si el término t es común o no en todos los documentos del corpus (D). Se obtiene dividiendo el número total de documentos por el número de documentos que contienen el término, y se toma el logaritmo de ese cociente:

$$\mathrm{idf(t, D)} = \log \frac{|\mathrm{D}|}{1 + |\{\mathrm{d} \in \mathrm{D} : \mathrm{t} \in \mathrm{d}\}|}$$

Cuanto menor es el número de documentos que contienen al término en relación con el tamaño del corpus, mayor es el factor. Se utiliza el logaritmo para suavizar la penalización de este factor.

Finalmente, tf-idf(t, d, D) se calcula como $\mathrm{tf(t, d)} \times \mathrm{idf(t, D)}$. Un valor alto en tf-idf se alcanza con una elevada frecuencia de término (en el documento dado) y una baja frecuencia de ocurrencia del término en la colección completa de documentos.

Esta métrica es típicamente utilizada para la extracción de *keywords* de un documento mediante la obtención del tf-idf de todos los n-gramas de este documento, considerando el top k como *keywords* . Sin embargo, no existen reglas definidas para la elección del n de los n-gramas o el k del top k. Además, cabe notar que se trata de una métrica muy dependiente del corpus con el que se trata, lo cual podría ser una desventaja en el caso de un corpus cambiante.

[3] Conjunto de textos.

[4] Subsecuencia de n elementos de una secuencia dada, en este caso palabras.

2.2.2. *Rapid Automatic Keyword Extraction* (RAKE)

El algoritmo de Extracción Rápida y Automática de *Keywords* (RAKE) fue propuesto el 2010 como un método de extracción de *keywords* eficiente, no supervisado, independiente del dominio e independiente del idioma [15]. RAKE permite operar sobre un documento de forma independiente, sin referirse a un corpus, a diferencia de otros métodos, como el uso de tf-idf, que a pesar de que funcionan bastante bien, tienen la desventaja de depender del estado actual del corpus utilizado, el cual podría potencialmente cambiar.

RAKE se basa en la observación de que las *keywords* típicamente contienen más de una palabra, pero rara vez contienen puntuaciones o palabras vacías (o *stopwords*) como artículos, preposiciones u otras palabras comunes, ya que estas suelen ser consideradas poco informativas. El algoritmo opera de la siguiente forma para encontrar las *keywords* de un documento:

Keywords candidatas

Se obtienen las *keywords* candidatas de la siguiente forma:

- Se divide el documento en cada delimitador de frase, tales como puntos, comas, signos de interrogación, entre otros signos de puntuación.
- Luego se divide cada frase en secuencias de palabras separadas por *stopwords*. Cada una de estas secuencias se considera como una *keyword* candidata.

Keywords contiguas

Como anteriormente se obtienen las *keywords* candidatas separando las frases por *stopwords*, las *keywords* obtenidas hasta ahora no contienen *stopwords*. Sin embargo, sí es posible encontrar frases en un texto que podrían ser consideradas "clave" y contienen *stopwords*, como, por ejemplo, "Internet of Things". Para encontrar estas frases, RAKE además busca pares de *keywords* contiguas que se repitan al menos dos veces en el mismo documento y en el mismo orden. En caso de encontrar algún caso, se agrega a la lista de candidatas la combinación de las palabras y sus *stopwords* interiores.

Asignación de puntajes

Una vez que se tienen las posibles *keywords*, se le asigna un puntaje a cada una, el cual se define como la suma de los puntajes de cada palabra que la conforma. A su vez, el puntaje de una palabra se calcula según la métrica:

$$\text{score}(\text{word}) = \frac{\deg(\text{word})}{\text{freq}(\text{word})}$$

Donde freq(word) es la frecuencia de la palabra, es decir, la cantidad de veces que ocurre esa palabra en la lista de candidatas, y deg(word) es el grado de la palabra, el cual representa la cantidad de co-ocurrencias de esta palabra con otras palabras en la lista de candidatos. Esta métrica favorece a palabras que se presentan en *keywords* más largas y desfavorece a palabras que se presentan con mucha frecuencia y en *keywords* más cortas.

Extracción de *keywords*

Luego de asignar un puntaje a las candidatas, se seleccionan las mejores T como *keywords* para el documento, donde T generalmente se fija como 1/3 de la lista de candidatas.

2.2.3. KEA

KEA (*Key-phrase Extraction Algorithm*) [16] es un algoritmo para extracción de *keywords* mediante una mezcla de métodos léxicos para seleccionar frases candidatas y aprendizaje automático para predecir cuáles candidatas son efectivamente buenas *keywords*, utilizando un clasificador Naive Bayes. Al igual que RAKE, KEA es independiente del lenguaje, aunque requiere una lista de *stopwords* y un *stemmer*[5], que dependen del idioma.

El algoritmo tiene dos fases:

- Entrenamiento, en el que se crea un modelo para identificar *keywords*, usando documentos para este propósito, para los cuales ya se cuenta con *keywords* identificadas por el autor.
- Extracción, en la que se eligen las *keywords* para un documento nuevo utilizando el modelo anterior.

KEA opera de la siguiente forma para encontrar las *keywords* de un documento:

Extracción de candidatos

Se extraen todas las frases del texto que cumplan con las siguientes condiciones:

- Frases limitadas a una longitud predefinida (generalmente tres palabras).
- Las frases no pueden ser nombres propios. Para esto se filtran palabras que solo se encuentran con mayúscula.
- Frases que no comienzan ni terminan con una *stopword*.

Todas las secuencias contiguas de palabras en cada línea del documento se prueban usando las tres reglas anteriores, produciendo un conjunto de frases candidatas. Además, cabe notar que sub-frases también suelen ser candidatas.

[5]Herramienta que permite reducir una palabra a su raíz.

Características

Para cada frase candidata se computan los siguientes valores:

- **Tf-idf:** métrica descrita anteriormente. Las frases candidatas que tienen un alto valor de tf-idf tienen más probabilidades de ser frases clave.
- **Primera aparición:** se calcula como el porcentaje de texto anterior a la primera aparición del término en el documento, con respecto al texto total. Es más probable que los términos que aparecen al principio o al final de un documento sean frases clave.
- **Longitud de una frase:** el número de sus palabras componentes.
- **Grado de la palabra:** la cantidad de frases en el conjunto de candidatos que están semánticamente relacionadas con esta frase. Las frases con alto grado tienen más probabilidades de ser frases clave.

Entrenamiento del modelo

Antes de poder extraer frases clave de documentos nuevos, KEA primero necesita crear un modelo que aprenda la estrategia de extracción de los documentos indexados manualmente. Dada la lista de frases candidatas, KEA marca las que se asignaron manualmente como ejemplos positivos y el resto como ejemplos negativos, y se utilizan las características definidas anteriormente para cada una para entrenar el modelo.

Extracción de *keywords*

Al extraer frases candidatas de documentos nuevos, KEA toma el modelo y los valores de las características para cada frase candidata y calcula su probabilidad de ser una frase clave. Las frases con las probabilidades más altas se seleccionan en el conjunto final de frases clave.

2.3. Aprendizaje automático

El aprendizaje automático es una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas para lograr que un computador tenga la habilidad de aprender a realizar una tarea, sin necesidad de ser programado explícitamente para esta, generalizando comportamientos a partir de datos de ejemplo. Estas técnicas típicamente se aplican a tareas computacionales en las que el diseño y la programación de algoritmos explícitos con un buen rendimiento es difícil o inviable.

Dentro del área de aprendizaje automático existen dos principales categorías. La primera de ellas es el aprendizaje supervisado, cuyo objetivo es crear un modelo que se aproxime a una función que mapea una entrada a una salida o etiqueta, basándose en datos de entradas de ejemplo con etiquetas conocidas [8]. En general esto se realiza en dos fases: el entrenamiento,

en el cual se cuenta con un sub-conjunto de datos de ejemplo etiquetados y el modelo se ajusta a estos datos, y una fase de pruebas, en la cual se cuenta con otro sub-conjunto de datos de ejemplo, distinto al anterior, y se evalúa el desempeño del modelo. También se puede agregar a esto una fase de validación.

El aprendizaje supervisado también incluye otros dos tipos de aprendizajes. El primero de ellos es el aprendizaje semi-supervisado, donde se cuenta con algunos datos etiquetados, pero no todos. El segundo de ellos es el aprendizaje por refuerzo, en el cual los datos de entrenamiento se dan solo como retroalimentación a las acciones del programa.

Por otra parte, está el aprendizaje no supervisado, en el cual no se asignan etiquetas al algoritmo de aprendizaje, dejándolo solo para encontrar estructura en su entrada. Uno de los enfoques utilizados para esta tarea es el de "clustering", el cual consiste en agrupar los datos de entrada de forma que los objetos de un grupo sean similares o relacionados entre sí y sean diferentes a los elementos de otros grupos [8].

En particular, existen dos tipos de aprendizaje supervisado, los cuales se describen a continuación.

2.3.1. Clasificación

Se refiere como clasificación al problema de identificar a qué clase, dentro de un conjunto definido de dos o más clases, pertenece una observación nueva. Dicho de otra forma, es la tarea de crear un modelo que se aproxime a la función que mapea variables de entrada con variables de salida discretos.

Es común que las predicciones de los modelos de clasificación se presenten como un vector de probabilidades, las cuales se pueden interpretar como las probabilidades de que un ejemplo dado pertenezca a cada una de las posibles clases. Una probabilidad pronosticada se puede convertir en un valor de clase seleccionando la etiqueta de clase que tenga la probabilidad más alta.

Métricas de desempeño

Hay varias maneras de estimar el desempeño de un modelo de clasificación, siendo una de las más comunes calcular la exactitud o *accuracy*. La exactitud en los problemas de clasificación es la cantidad de predicciones correctas hechas por el modelo sobre todas las predicciones hechas. Es una buena métrica cuando los valores de salida (las clases) de los datos de ejemplo están equilibradas. En el caso contrario, un modelo puede predecir el valor de la clase mayoritaria para todas las predicciones y lograr una alta precisión de clasificación, pero este modelo no sería útil en el dominio del problema. Estos casos demuestran que la exactitud por sí sola no es información suficiente para decidir si el clasificador es o no un buen modelo.

La precisión es la proporción de cuánto de lo predicho es correcto. En el caso de un

clasificador binario, se refiere a la cantidad total de predicciones positivas correctas dividido por el total de predicciones positivas, correctas o incorrectas.

Por otra parte, la recuperación o *recall*, en el caso de un clasificador binario, es el número de predicciones positivas dividido por el número total de valores de clase positivos en los datos de prueba. Se puede pensar como una medida de la "completitud" del clasificador.

2.3.2. Regresión

Se habla de regresión cuando se trata con variables de salida reales, como cantidades o valores, es decir, el objetivo es crear un modelo que se aproxime a la función que mapea de las variables de entrada a valores de salida continuos. Los modelos de regresión se basan en el análisis de las relaciones entre variables y tendencias con el fin de hacer predicciones sobre variables continuas.

Métricas de desempeño

Al igual que en el caso de la clasificación, existen varias métricas para evaluar un modelo de regresión. Una de ellas es el error absoluto medio, que se define como la suma de las diferencias absolutas entre las predicciones y los valores reales. Esta medida da una idea de cuán equivocadas están las predicciones, pero no de la dirección del error, el cual podría estar por encima o por debajo de la predicción.

El error cuadrático medio mide el promedio de los cuadrados de los errores. Es muy parecido al error absoluto medio, ya que proporciona una idea general de la magnitud del error. También se puede tomar la raíz cuadrada del error cuadrático medio, lo cual convierte las unidades a las unidades originales de la variable de salida.

2.3.3. Modelos de aprendizaje supervisado

A continuación se describen algunos modelos supervisados, tanto de clasificación como regresión.

Support Vector Machines (SVM)

Las máquinas de vectores de soporte o *Support Vector Machines* (SVM), son modelos de aprendizaje supervisados no probabilísticos. Fueron desarrollados para problemas de clasificación binarios, aunque las extensiones de la técnica se han realizado para admitir problemas de regresión y clasificación de clases múltiples.

Dado un conjunto de ejemplos de entrenamiento, un modelo SVM es una representación de los ejemplos como puntos en el espacio, mapeados de manera que los ejemplos de las

categorías separadas estén divididos por un espacio libre lo más ancho posible. Luego, se mapean nuevos ejemplos en ese mismo espacio y se predice que pertenecen a una categoría según el lado del espacio en el que caen. Esto se hace usando un proceso de optimización que solo considera aquellas instancias de datos en el conjunto de datos de capacitación que están más cerca de la línea que mejor separa las clases. Las instancias se llaman "vectores de soporte".

SVM se desarrolló para variables de entrada numéricas, aunque se puede convertir valores nominales en valores numéricos. Además, los datos de entrada deberían ser normalizados antes de ser utilizados, para obtener mejores resultados [5]. La principal ventaja de escalar los datos de entrada es evitar los atributos en mayores rangos numéricos que dominan aquellos en rangos numéricos más pequeños [7].

En general, no es posible trazar una línea para separar cuidadosamente las clases, por lo tanto se agrega un margen alrededor de la línea para relajar la restricción, permitiendo que algunas instancias se clasifiquen erróneamente pero con un mejor resultado general. Por otra parte, pocos conjuntos de datos se pueden separar con solo una línea recta. En ocasiones, se debe marcar una línea con curvas o incluso regiones poligonales.

Naive Bayes

Naive Bayes es una familia de algoritmos de clasificación probabilísticos basados en el Teorema de Bayes, el cual describe la probabilidad de un evento, basado en el conocimiento previo de las condiciones que podrían estar relacionadas con el evento. En los algoritmos naive Bayes, la probabilidad previa para cada clase se calcula a partir de los datos de entrenamiento y se supone que todas las características o *features* de los datos de entrada son independientes entre sí, supuesto al que se debe el nombre de "naive" o "ingenuo" en español. Este supuesto es poco realista y muchas veces no se cumple, pero hace que las probabilidades sean rápidas y fáciles de calcular. A pesar de esto, con el procesamiento previo apropiado, naive Bayes ha demostrado ser competitivo en este dominio con métodos más avanzados, incluidas máquinas de vectores de soporte [14].

Regresión lineal

La regresión lineal es un modelo de regresión muy simple, donde la variable dependiente o el valor de salida es una combinación lineal de los parámetros. Se representa como una ecuación lineal que combina un conjunto específico de valores de entrada (x) cuya solución es la salida pronosticada para ese conjunto de valores de entrada (y). La ecuación lineal asigna un factor de escala o coeficiente (β) a cada valor de entrada.

Por ejemplo, en el caso de una regresión de un valor de entrada y un valor de salida, la forma del modelo sería: $y = \beta_0 + \beta_1 x$. El concepto detrás de esto es: dado puntos de datos de ejemplo, intentar trazar una línea que se aproxime a los puntos de la mejor manera. En el caso de que se tenga más de un valor de entrada, la línea pasa a ser un hiperplano.

2.4. *Learning to rank*

Learning to rank se refiere a las técnicas de aprendizaje automático para entrenar un modelo en una tarea de ordenamiento. Estas técnicas se pueden aplicar en una amplia variedad de tareas, entre ellas, recuperación de documentos, motores de búsqueda, traducción automática y extracción de *keywords* [10].

Learning to rank es una tarea de aprendizaje supervisado, por lo cual se cuenta con una fase de entrenamiento y una fase de pruebas. Los datos de entrenamiento consisten en listas de elementos que pertenecen a una misma consulta o *query*, con algún orden especificado entre los elementos de cada lista. En general este orden está especificado por etiquetas, que representan el nivel o puntaje del elemento. El objetivo es producir una permutación de elementos en listas nuevas de una manera que sea similar a las clasificaciones de los datos de entrenamiento.

Existen tres principales enfoques:

Enfoque por puntos

En el enfoque por puntos se ignora la estructura de ordenamiento por grupo o *query*, tomando un solo elemento a la vez. Se pueden aplicar métodos existentes para clasificación, regresión o clasificación ordinal, entrenándolos para predecir qué tan relevante es cada elemento de forma independiente. El ranking final se logra simplemente ordenando la lista de resultados por estos puntajes de documentos [10].

Enfoque por pares

El enfoque por pares toma un par de elementos a la vez, intentando encontrar el ordenamiento óptimo para cada par. El objetivo del ordenamiento es minimizar el número de inversiones en el orden, es decir, los casos en que un par de resultados está en el orden equivocado. En el enfoque por parejas, la estructura del ordenamiento de cada grupo también se ignora, no tomando en consideración cuando un par se genera a partir de elementos asociados a la misma *query* [11].

En general, el ordenamiento se plantea como una clasificación binaria. Para el entrenamiento se recopilan pares de elementos de los datos de ejemplo y para cada par de elementos se asigna una etiqueta que representa la relevancia relativa de los dos documentos. Por ejemplo, dado un par de elementos (e_1, e_2), si e_1 es mejor que e_2 entonces se le asigna la etiqueta 1 a ese par, caso contrario, si e_2 es mejor que e_1 se le asigna la etiqueta -1 al par. Luego se entrena un modelo de clasificación con los pares de datos etiquetados, donde el clasificador se entrena para predecir, dado un par de elementos, cuál es el orden entre ellos [3]. Dicho de otra forma, dado un par de elementos, el modelo entrenado permite predecir cuál de ellos es superior, lo cual, a su vez, permite definir un orden de toda la lista de elementos.

El algoritmo Ranking SVM[11] es un ejemplo de un método que sigue este enfoque, el cual usa SVM como algoritmo de clasificación. También, LambdaMART [2] sigue esta técnica, además aplicando ciertas optimizaciones al proceso.

Enfoque por listas

En el enfoque por listas se considera la lista completa de elementos que pertenecen a una misma *query*, intentando encontrar el orden óptimo para esta. En teoría, este enfoque funciona mejor que los dos anteriores, ya que es el que más se asemeja al problema de ordenamiento real [11]. Sin embargo, esto significa una mayor complejidad en los métodos que siguen el enfoque por listas. Algunos algoritmos que siguen este enfoque son ListNet [3] y AdaRank [17].

2.4.1. Métricas de evaluación

La evaluación del rendimiento de los modelos de ordenamiento se realiza mediante una comparación entre el ordenamiento de una lista de elementos generado por el modelo y el orden ya conocido de esos elementos. Algunas métricas utilizadas para esto son las siguientes.

Ganancia acumulada descontada (DCG)

La ganancia acumulada descontada o *Discounted Cumulative Gain* (DCG) consiste en la suma de los valores de relevancia o puntajes de todos los elementos en la lista, con una penalización del logaritmo de la posición del elemento al que corresponde el valor:

$$\mathrm{DCG} = \sum_{\mathrm{i}=1}^{\mathrm{l}} \frac{\mathrm{score_i}}{\log_2(\mathrm{i}+1)}$$

Donde l es el largo de la lista y $\mathrm{score_i}$ es el puntaje del elemento i. Esta métrica se basa en la premisa de que los elementos más relevantes que aparecen más abajo en un ordenamiento deberían penalizarse [10]. Además, se puede calcular la ganancia acumulada descontada considerando solo los top n elementos, en cuyo caso se reemplaza la l por el valor n y se denota DCG@n.

Ganancia acumulada descontada normalizada (NDCG)

La ganancia acumulada descontada normalizada o *Normalized Discounted Cumulative Gain* (NDCG) consiste en la ganancia acumulada descontada normalizada por la ganancia acumulada ideal (IDCG), la cual se calcula usando el ordenamiento ideal de los elementos:

$$\text{nDCG} = \frac{\text{DCG}}{\text{IDCG}}$$

Esta métrica es útil para comparar el rendimiento de un modelo para datos de distintas *queries*, ya que la cantidad de elementos en cada una es variable [10]. Al igual que en el caso anterior, se puede calcular la ganancia acumulada descontada considerando solo los top n elementos, en cuyo caso se calcula DCG@n y IDCG@n y se denota como NDCG@n.

Tau de Kendall

El coeficiente de correlación de ordenamiento de Kendall, comúnmente conocido como el coeficiente tau de Kendall, es una métrica utilizada para medir la asociación de orden entre dos listas. La correlación de Kendall entre dos variables será cercana a 1 cuando las observaciones tengan un orden similar y cercana a 0 cuando las observaciones tienen un orden diferente. Por otra parte, valores cercanos a -1 indican una asociación negativa o un orden inverso.

Existen distintos tipos de coeficientes de Kendall, los cuales varían en la forma en la que tratan con los empates dentro de los ordenamientos. En el caso del tipo tau-a (τ_a) no se hace un procedimiento especial en el caso de haber empates, pero los tipos tau-b (τ_b) y tau-c (τ_c) tratan con ellos de formas distintas. En particular el tipo τ_b se calcula de la siguiente forma:

$$\tau_b = \frac{(P - Q)}{\sqrt{(P + Q + T) * (P + Q + U)}}$$

Donde, dado dos ordenamientos x e y de una lista, P es la cantidad de pares concordantes, Q es la cantidad de pares discordantes, T es la cantidad de empates ocurridos en x y U es la cantidad de empates ocurridos en y. Si un empate ocurre para el mismo par tanto en x como en y, este no se agrega a T y U.

2.5. Resumen

En este capítulo se describen los conceptos y herramientas principales involucrados en este trabajo de memoria. En primera instancia se describen las *tag clouds*, formato de visualización escogido para este trabajo, en particular las ventajas y desventajas de su uso. En seguida, se describen las técnicas de extracción de *keywords* consideradas para la tarea de poblar las *tag clouds*: tf-idf, RAKE y KEA. Luego, con el propósito de definir un orden de relevancia para las *keywords*, requerido para la visualización final, se acude a técnicas de aprendizaje automático, las cuales son descritas de forma general. Finalmente, se opta por técnicas de aprendizaje automático para una tarea de ordenamiento, referidas como *learning to rank*, para lo cual se describen los tres principales enfoques: por puntos, por pares y por listas.

Además, se muestran tres métricas de evaluación de modelos de *learning to rank*: DCG, NDCG y tau de Kendall, las cuales son utilizadas para evaluar los modelos entrenados.

Capítulo 3

Solución propuesta

En este capítulo se describe el sistema propuesto para la creación de *tag clouds*. En primer lugar se describe de forma general la arquitectura de la herramienta. Luego, se muestran a fondo cada una de las etapas del proceso completo dado el identificador de un académico, desde la extracción de sus publicaciones hasta la creación de las visualizaciones mismas. Finalmente, se describe el servicio web creado para el uso de la herramienta.

3.1. Arquitectura de la solución

El proceso completo propuesto para la creación de *tag clouds* de un académico consiste de cuatro grandes tareas: la obtención de las publicaciones, la extracción de *keywords* de estas, la selección de las mejores *keywords* y finalmente la generación de las *tag clouds*. Cada una de estas tareas se subdivide en módulos, los cuales se pueden observar en la figura 3.1.

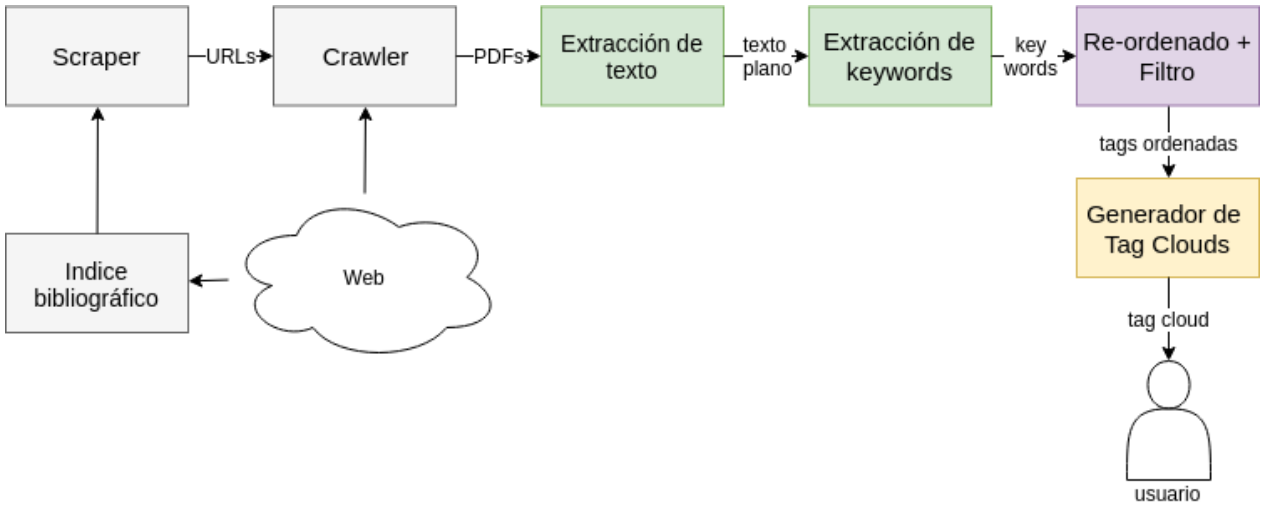

Figura 3.1: Arquitectura de la solución propuesta

En primer lugar, se necesita obtener los URLs de cada una de las publicaciones, los cuales luego pueden ser descargados mediante un *crawler*. Dado que las publicaciones generalmente se encuentran en formato PDF, antes del proceso de extracción de *keywords* se requiere extraer el texto plano de estos archivos. Después de esto se pueden extraer las *keywords* candidatas a formar parte de la visualización final. Luego de esto, hay que realizar la selección de las *keywords*, para lo cual se requiere un re-ordenamiento de las candidatas, permitiendo definir cuáles son las mejores y descartar las demás. Finalmente, con esta información se crean las *tag clouds*. En las secciones siguientes se describe a fondo cada uno de estos módulos, además de las herramientas y técnicas escogidas para llevarlos a cabo.

3.2. Extracción de las publicaciones

Existen muchas editoriales de revistas académicas, varias de las cuales cuentan con su propia página web en la cual se pueden encontrar sus publicaciones en formato digital. Algunos ejemplos son ACM, Springer, Elsevier, entre otras. Es por esto que herramientas como Google Scholar y DBLP, en el área de Ciencias de la Computación, facilitan bastante el proceso de búsqueda de publicaciones académicas, ya que estas herramientas permiten realizar búsquedas de artículos publicados por distintas editoriales, proporcionando el link a la versión electrónica de la publicación hospedada en la página de la editorial. Además, en ambas herramientas las búsquedas están indexadas por autor, lo cual agiliza la obtención de todas las publicaciones de un autor. Aunque muchas no están disponibles de forma gratuita, varias universidades, incluyendo la Universidad de Chile, cuentan con suscripciones que permiten el acceso a algunas publicaciones dentro del *paywall*[1] de la universidad.

Debido a esto, la extracción de las publicaciones se divide en dos principales tareas: la obtención del link a la versión electrónica de la publicación y la extracción del PDF de dicha página. Ambas tareas se describen a continuación, además de la elección de la fuente de información para la primera de estas.

3.2.1. Fuente de información

Como trabajo previo, es necesario decidir una fuente de información para la obtención de las versiones electrónicas de las publicaciones. Como el objetivo planteado se refiere en un principio a perfiles de investigadores del área de Ciencias de la Computación, la primera opción sería usar como fuente DBLP, debido a su especialización en el área y la mayor confiabilidad de los recursos que se encuentran en este repositorio.

A pesar de esto, también existe la posibilidad de usar Google Scholar, ResearchGate o Academia.edu para este fin. Sin embargo, estas son descartadas debido a tres razones. En primer lugar, ResearchGate y Academia.edu, debido a que están más planteadas como redes

[1] *Paywall* o "muro de pago", es un sistema que restringe el acceso a sitios web a usuarios que no cuentan con una suscripción pagada. En este caso, se refiere al acceso que se permite a estas páginas al conectarse a la red de la universidad.

sociales que como repositorios bibliográficos, se podría dificultar la extracción automática de las versiones electrónicas de las publicaciones. Además, tienen menos publicaciones y perfiles que DBLP o Google Scholar en el área de Ciencias de la Computación, lo cual se verifica revisando publicaciones de académicos del Departamento de Ciencias de la Computación de la universidad.

Por otra parte, existe una preferencia al uso de DBLP por sobre Google Scholar, ya que se considera más "confiable" en su información. Esto es debido a que hay varios casos de ambigüedad entre académicos con el mismo nombre en este último, existiendo una confusión entre las publicaciones de estos. Además, la selección de publicaciones en Google Scholar no es tan minuciosa como en repositorios bibliográficos más tradicionales, por lo cual artículos sin validación pueden terminar ahí.

Finalmente, dado que DBLP se encuentra disponible en un formato RDF bastante fácil de procesar, lo cual provee una alternativa que puede facilitar el proceso de extracción de las publicaciones, y, por otra parte, que se quiere limitar el alcance de la herramienta a investigadores de Ciencias de la Computación, se opta por este repositorio bibliográfico como la mejor fuente de información para la herramienta.

3.2.2. Acceso a DBLP

Se exploran dos alternativas para la extracción de las publicaciones de DBLP.

Endpoint de SPARQL

La primera alternativa es el uso de un *endpoint*[2] de SPARQL de DBLP[3], el cual usa como fuente la base de datos publicada con el Servidor D2R[4] de DBLP. Esto, mediante una simple consulta, permite la obtención de todas las publicaciones de un académico con la respectiva *homepage* o página principal de la publicación en sitios como ACM, SpringerLink, ScienceDirect, entre otros, donde en la mayoría de los casos se puede encontrar una versión descargable en formato PDF dentro del *paywall* de la universidad.

Para poder usar el *endpoint* en Python se quiere utilizar la librería SPARQLWrapper[5], la cual permite ejecutar remotamente consultas a un servicio SPARQL y ayuda a convertir el resultado a un formato más manejable.

Un problema que se descubre de esta alternativa es la inconsistencia de la disponibilidad del *endpoint*, el cual se ha encontrado caído durante periodos de tiempo prolongados. Es por esto que se explora el uso del formato RDF de DBLP como un enfoque alternativo.

[2]Punto de entrada a un servicio, en este caso a un servicio web.
[3]http://dblp.l3s.de/d2r/sparql
[4]Herramienta para publicar el contenido de bases de datos relacionales en la Web Semántica.
[5]https://github.com/RDFLib/sparqlwrapper

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix ns0: <https://dblp.uni-trier.de/rdf/schema-2017-04-18#> .
...
<https://dblp.uni-trier.de/pers/h/Hogan:Aidan>
  owl:sameAs <https://dblp.org/pers/h/Hogan:Aidan> .
<https://dblp.org/pers/h/Hogan:Aidan>
  a <https://dblp.uni-trier.de/rdf/schema-2017-04-18#Person> ;
  ns0:primaryFullPersonName "Aidan Hogan" ;
  ns0:primaryAffiliation "Universidad de Chile" ;
  ns0:primaryHomepage <http://aidanhogan.com/> ;
  ns0:otherHomepage <http://sw.deri.org/~aidanh/> ;
  ns0:authorOf <https://dblp.org/rec/conf/amw/GonzalezH18>,
    <https://dblp.org/rec/conf/amw/Rosales-MendezP18>,
    <https://dblp.org/rec/conf/www/GonzalezH18>,
    <https://dblp.org/rec/conf/www/Hogan18>,
    <https://dblp.org/rec/conf/www/FerradaBBH18>,
    <https://dblp.org/rec/conf/www/SaezH18>,
    <https://dblp.org/rec/journals/biomedsem/KhanSMHMRS17>,
    <https://dblp.org/rec/journals/csur/AnglesABHRV17>,
    <https://dblp.org/rec/journals/semweb/VandenbusscheUM17>,
    <https://dblp.org/rec/journals/tweb/Hogan17>,
    <https://dblp.org/rec/conf/semweb/Rosales-MendezP17>,
    <https://dblp.org/rec/conf/semweb/FerradaBH17>,
    ...
```

Figura 3.2: Ejemplo de un archivo en formato RDF (extracto)

Formato RDF

La segunda alternativa, la cual es finalmente escogida, es el uso de la API de exportación explícita de la página de cada académico. Para cada página de persona en DBLP se puede acceder a su información directamente en diferentes formatos de exportación, en particular formatos RDF, N-Triples y RDF/XML. A su vez, estos archivos en formato RDF contienen triples que indican todos los artículos que ha escrito el académico, de la forma `[persona] authorOf [paper]`. En la figura 3.2 se puede ver un extracto de un archivo de este tipo[6].

A partir de cada uno de estos triples se puede obtener el link a la versión descargable de la descripción en RDF del artículo, donde se puede encontrar el link a la edición electrónica de la publicación de la forma `[paper] primaryElectronicEdition [link]`. Además, se puede encontrar el año de publicación, título y descripción del paper, entre otros.

De esta forma, se utiliza la librería RDFLib[7] de Python para parsear estos archivos estructurados y obtener la información necesaria, es decir, dado un académico, obtener una lista con todos los links a las versiones electrónicas de las publicaciones de un académico.

[6]Se muestra un archivo con sintaxis Turtle debido a su forma más compacta y fácil de leer, sin embargo, los archivos utilizados están en sintaxis N-triples.

[7]https://rdflib.readthedocs.io/

3.3. Extracción de PDFs

Se requiere obtener las publicaciones de perfiles de DBLP, donde cada una cuenta con un link a la página publicadora, como por ejemplo ACM Digital Library[8], que es una de las más frecuentes, la cual a su vez cuenta con un link de descarga al PDF, en el caso de estar en una máquina dentro del *paywall* de la universidad.

Una vez obtenido el link a la versión electrónica de una publicación, para descargar el archivo en formato PDF de forma automática se requiere hacer un *scraping* a la página web en la que se encuentra hospedado. Para esto se plantea el uso del *framework* de *web crawling* para Python, Scrapy[9]. Este se basa en *spiders* o *crawlers*[10] autónomos a los que se les da un conjunto de instrucciones. Es necesario que esto se corra dentro del *paywall* de la universidad, para poder efectivamente obtener el link de descarga.

Para el proceso de *scraping* se observa que existen cuatro principales casos:

1. El link a la versión electrónica consiste en un link directo al PDF, sin tener que pasar por otra interfaz, en cuyo caso basta con descargarlo.
2. El PDF se encuentra en una página estática con un link de descarga visible, como se muestra en la figura 3.3. En este caso el proceso es simple, ya que basta con encontrar el link de descarga mediante un *spider* de Scrapy. Un problema encontrado en este caso es que muchas veces existe más de un link de descarga a PDF en la página, por ejemplo el link al artículo y el link a la revista completa, por lo cual es necesario diferenciar estos casos.

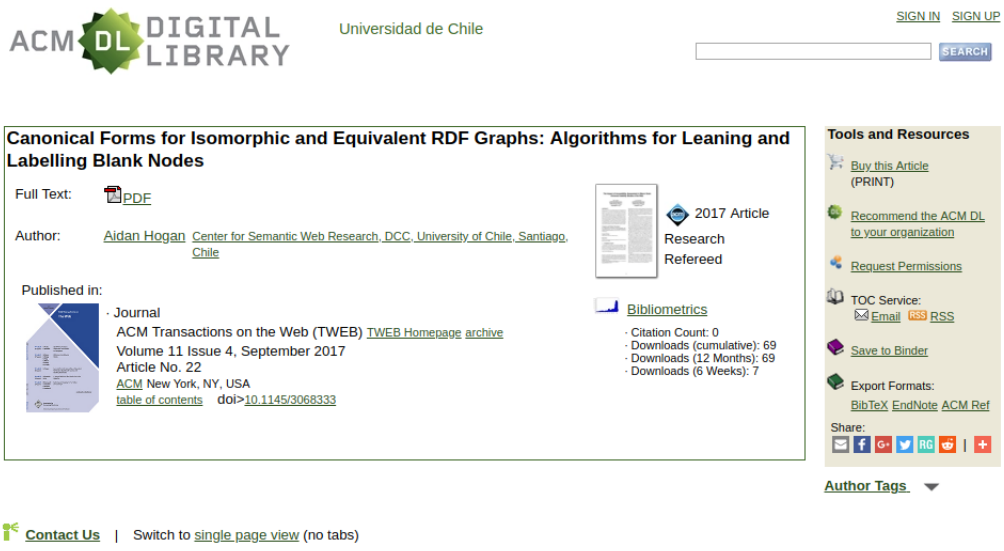

Figura 3.3: Ejemplo de una página estática con link visible

[8]https://dl.acm.org/
[9]https://scrapy.org/
[10]Aplicación que navega por la web de forma automática, en este caso con el propósito de extraer información de ciertas páginas web.

3. El PDF se encuentra en una página dinámica o en un link de descarga accesible mediante clicks a menús generados de forma dinámica con JavaScript. En este caso no basta con el simple uso de Scrapy, sino que se requiere el uso de una herramienta que permita cargar la página en un navegador. Dado que existe una cantidad no menor de publicaciones hospedadas en páginas de este tipo, se considera el uso de Selenium para Python[11] en estos casos. Selenium es un framework diseñado para realizar pruebas automáticas para aplicaciones web, pero que puede ser bastante útil cuando se trata de realizar *scraping*. Sin embargo, el proceso de extracción mediante el uso de Selenium es considerablemente más costoso que el anterior, llegando a demorarse alrededor de 15 minutos en extraer 10 PDFs, cuando en el caso anterior se demora menos de 1 minuto. Es por esto que este proceso se realiza como última opción para las publicaciones no extraídas con los dos métodos anteriores, en el caso de que la tasa de publicaciones extraídas con respecto al total sea muy baja, menor a un 50 %.

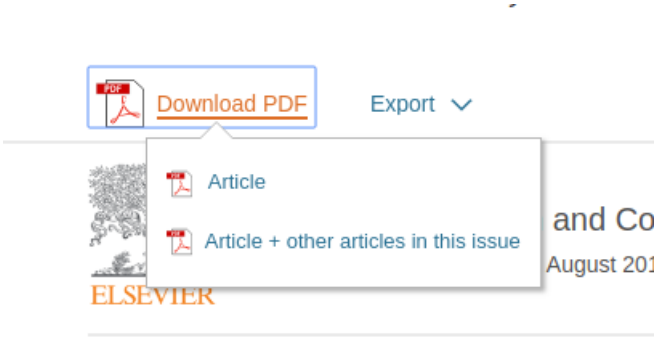

Figura 3.4: Ejemplo de una página estática con menú generado con JavaScript (Science Direct)

4. No se tiene acceso al PDF en el *paywall* de la universidad, en cuyo caso se descartan estas publicaciones.

Cabe mencionar que el objetivo de este trabajo no es la re-publicación de los artículos extraídos, tanto en parte como en su totalidad, sino que utilizar este contenido para generar visualizaciones de utilidad.

3.4. Extracción de *keywords*

Una vez obtenidas las publicaciones en formato PDF, en primer lugar, para trabajar con el contenido de estas se extrae el texto, del cual luego se pueden extraer *keywords* con los métodos mencionados en 2.2. Se observa que algunos papers tienen una sección en la cual los autores definen las *keywords* del texto, sin embargo, dado que la cantidad de textos que siguen este formato no es predominante, es decir, no todos los autores declaran las *keywords*, no se considera como posible método de extracción de *keywords* obtenerlas de esta forma.

A continuación se describen las tareas de *parsing* y extracción de *keywords* del texto.

[11] http://selenium-python.readthedocs.io/

3.4.1. Extracción del texto y pre-procesamiento

El propósito original del formato PDF era ser un formato neutral, independiente del software, hardware o sistema operativo, para preservar los datos y la presentación de un documento, principalmente para documentos impresos. Como su foco es la presentación del documento, no se preocupa mucho por la estructura y no contiene *tags* para identificar los distintos elementos del texto. Para identificar la estructura de un PDF se requiere recorrer todo el texto e identificar los elementos según las propiedades del texto. Además, para identificar párrafos, líneas y palabras es necesario observar el posicionamiento de los fragmentos de texto y los espacios entre las palabras, entre otras cosas.

Para facilitar el proceso de extracción del texto de los archivos PDF, se utiliza la librería para Python PDFMiner.six[12], la cual logra extraer el texto de archivos PDF con bastante precisión. Aún así, se requiere una modificación del texto extraído en algunos casos, en particular algunos caracteres especiales, ligaduras tipográficas[13] y las palabras separadas por guión cuando no calzan dentro de una línea.

Finalmente, cabe notar que no se consideran los casos en que el contenido de un PDF no es texto sino que imagen, como el caso de documentos escaneados, ya que estos son menos comunes y requerirían del uso de reconocimiento óptico de caracteres.

3.4.2. Extracción de *keywords* según su tf-idf

La primera técnica utilizada para la extracción de *keywords* es el uso de tf-idf, mediante la implementación perteneciente a la librería Scikit-learn[14] para Python. Esto se realiza considerando todas las publicaciones de un autor como un documento y, ya que se requiere de un corpus, se utilizan los documentos de 21 autores (profesores del DCC). Para la extracción de *keywords* de un autor se toman todas las frases de hasta 3 palabras de su documento correspondiente, principalmente porque *keywords* con más palabras ocupan demasiado espacio en una *tag cloud*, calculando su tf-idf con respecto al corpus disponible.

Los resultados obtenidos son mixtos. Esta metodología en general parece favorecer a frases cortas, típicamente de una palabra, y de esta forma se pierden las *keywords* compuestas. Por otra parte, también se encuentra entre las frases con mejores puntajes bastantes palabras del lenguaje común, pero que no son tan frecuentes, como "*credibility*", "*awareness*" o "*collaboration*", y que no ayudan a representar la información más relevante de los textos.

[12] https://github.com/pdfminer/pdfminer.six

[13] Una ligadura tipográfica ocurre cuando dos o más letras se unen como un solo glifo. Un ejemplo de esto es el uso del glifo "fi" en vez de "`fi`".

[14] http://scikit-learn.org/

3.4.3. Extracción de *keywords* con RAKE

Luego de observar estos resultados, se utiliza una segunda técnica de extracción de *keywords*, RAKE. Para esto se utiliza una implementación en Python del algoritmo, RAKE-tutorial[15], considerando *keywords* candidatas de a lo más tres palabras, por la razón mencionada anteriormente. A esta implementación se le realizan las siguientes modificaciones:

1. Se filtran los URLs para que estos no puedan ser seleccionados como parte de una *keyword*.
2. Se modifica la selección de *keywords* candidatas para que tome en consideración palabras compuestas como "*stop-word*" o "*pre-processing*" como una palabra, no como palabras separadas.
3. Se aplica una normalización a los sustantivos, transformando los plurales a su forma singular.

Los resultados obtenidos al correr el algoritmo para una publicación en general son mixtos; algunos son bastante acertados, pero también se encuentran frases irrelevantes o demasiado específicas. Sin embargo, dado que lo que se quiere obtener son *keywords* de todas las publicaciones, no solo una, se prueba el algoritmo sobre todas las publicaciones, considerando como puntaje final de cada *keyword* la suma de los puntajes parciales obtenidos en cada publicación en la que se encuentre, descrito en el capítulo anterior. Esto se realiza bajo el supuesto de que las frases más acertadas se deberían repetir en más de un paper, por lo que acumularían un puntaje mayor, y las más específicas o irrelevantes no se deberían repetir a lo largo de todos los papers, por lo que no acumularían un puntaje tan alto.

En la práctica, aunque efectivamente se obtienen *keywords* consideradas como "buenas" en el top 100, también se encuentran palabras muy comunes en este tipo de textos como "*figure*", "*case*", "*work*", entre otras. Por otra parte, efectivamente se logra filtrar algunas frases más específicas o que no tienen mucho sentido por si solas. Esto se discute más en profundidad en la sección 4.1, en particular cómo se decide si una *keyword* es buena o no.

3.4.4. Extracción de *keywords* con KEA

De forma paralela, se considera el uso del algoritmo supervisado KEA, para el cual se requiere de un set de entrenamiento. Inicialmente, se intenta utilizar como datos de entrenamiento las *keywords* definidas por autores incluidas en algunas publicaciones. Sin embargo, esto tiene ciertas desventajas, la primera de estas siendo que no todos los papers tienen esta sección. Otra desventaja encontrada es que dada la complejidad de parsear un PDF, además de que, aunque siguen una estructura similar, las estructuras de los papers pueden variar, la tarea de extraer la sección donde se encuentran las *keywords* no es trivial.

Por las razones anteriores y el hecho de que *a priori* no se contaba con otra forma de obtener un set de entrenamiento con facilidad, se descarta la posibilidad de usar KEA para la extracción de *keywords*. Aún así, se rescatan ciertas ideas implementadas en este método,

[15] https://github.com/zelandiya/RAKE-tutorial

en particular el uso de un enfoque de aprendizaje automático y algunas de las características usadas, las cuales se detallan en la sección 3.5.1.

3.4.5. Enfoque escogido

Basado en los resultados anteriores, los cuales por si solos no eran suficientemente buenos, se cambia un poco el enfoque inicial de la obtención de *keywords* solo mediante métricas. A cambio, se opta por experimentar con una extracción de *keywords* candidatas y posterior selección de las mejores mediante métodos de aprendizaje automático. Para la extracción se prefiere el uso de RAKE con la metodología descrita, obteniendo las mejores 500 *keywords* candidatas, dado el tipo de resultados obtenidos, los cuales favorecen más las frases que consisten de más palabras. Aún así, se utilizan ambos puntajes, RAKE y tf-idf, en la selección de *keywords*.

3.5. Selección de *keywords*

Para la tarea de selección de las mejores *keywords* mediante aprendizaje automático, en primer lugar, dado que se trata de texto, se plantean características que permitan representar las *keywords* para el modelo. Además, se muestran las dos formas distintas en las que se plantea el problema, clasificación y *learning to rank*.

3.5.1. Características

Se proponen las siguientes características para representar a cada *keyword*:

Métricas

- **RAKE:** se utiliza la suma del puntaje obtenido en cada paper mediante el algoritmo RAKE, descrita anteriormente.
- **Tf-idf:** se utiliza tf-idf, bajo el supuesto de que desfavorece a candidatas que corresponden a palabras comunes en el dominio tratado, complementando a RAKE de esta forma. Esta característica también es utilizada en KEA.
- **Profundidad de frase:** se plantea la hipótesis de que los términos que aparecen al principio de un documento, en el título o resumen del paper, son más relevantes y por lo tanto es más probable que sean *keywords*. Basado en esto, se propone calcular la profundidad de la aparición de una *keyword* en un paper como:

$$1 - \frac{\text{posición de la aparición}}{\text{total de palabras}}$$

Donde se tiene un decrecimiento lineal, por ejemplo, la profundidad de la primera palabra es 1 y la de una palabra que se encuentra en la mitad es 0,5. Además, dado

que probablemente una *keyword* aparece más de una vez, la profundidad total de la *keyword* se calcula como la suma de cada aparición.

- **Proporción de apariciones:** se calcula la proporción entre los documentos en los que se incluye la *keyword* y la cantidad total de documentos del autor.

Temporales

- **Año de primera aparición:** junto con la extracción del link a la versión electrónica de cada publicación, también se extrae su año de publicación. Con esta información se obtiene el año en el que fue publicado el primer paper que hace mención a cada *keyword*.
- **Año de aparición más reciente:** de la misma forma que la característica anterior, se obtiene el año en el que fue publicado el último paper que hace mención a cada *keyword*. Ambas características se incluyen dada la noción de que el área de investigación de un académico puede cambiar a través del tiempo y que podrían encontrar irrelevante una *tag cloud* con *keywords* relacionadas con áreas a las cuales ya no se dedican.
- **Años de apariciones:** en este caso se refiere a la diferencia entre el año más reciente y el primer año de aparición.

Otras

- **¿Está en wikipedia?:** se verifica si la *keyword* se puede encontrar como título de un artículo en Wikipedia, en cuyo caso se le asigna el valor 1, o no, en cuyo caso se le asigna el valor 0. Esta información se extrae de un volcado de la base de datos de Wikipedia[16].
- **Longitud de la frase:** es el número de sus palabras componentes, al igual que el usado en KEA.

Dada la varianza de los valores de las características, se aplica una estandarización a estos datos. Los valores se centran alrededor de cero y se ajustan para que tengan una varianza en el mismo orden de magnitud. Esto se realiza principalmente dado que muchos modelos podrían comportarse mal con datos no estandarizados, uno de ellos siendo SVM, como se menciona en 2.3.3.

3.5.2. Clasificación

En un comienzo se plantea el problema como una clasificación, donde se tienen clases numeradas del 1 al 5, siendo 1 la peor y 5 la mejor, y cada *keyword* puede pertenecer a una clase. Las técnicas utilizadas se describen a continuación.

[16] https://dumps.wikimedia.org/enwiki/latest/

Clasificación estándar

Como primera técnica, se ignora la naturaleza ordinal de las clases completamente y se trata a las 5 clases como valores desordenados. Para esto se utilizan modelos de clasificación típicos como *Support Vector Machine* y Naive Bayes, nuevamente ocupando la implementación de la librería Scikit-learn. Sin embargo, este acercamiento tiene la desventaja de que está ignorando la información que proporciona el ordenamiento de estas clases, donde una *keyword* clase 5 es mejor que una clase 4, la cual potencialmente podría mejorar el rendimiento predictivo de un clasificador.

Como alternativa, también se plantea como una clasificación binaria, cuyo propósito es decidir si una *keyword* pertenece a una clase mayor o menor a cierto umbral. Aunque de esta forma se logra definir con cierta precisión si una *keyword* es "aceptable" o no, donde la definición de "aceptable" es incierta, está lejos de solucionar el problema de selección de *keywords* para poblar una *tag cloud*.

Clasificación ordinal

Con el propósito de tomar en consideración el orden de las clases, se intenta seguir el enfoque descrito por Frank y Hall [4]. El método descrito convierte el problema de clasificación ordinal original en una serie de problemas de clasificación binaria que codifican el orden de las clases originales. Los datos se transforman de un problema ordinal de 5 clases a 4 problemas de clasificación binarios, donde, a modo de ejemplo, el primero de ellos sería un clasificador para definir si una entrada pertenece a una clase > 1 o no, el segundo si pertenece a una clase > 2, etc.

En la práctica, este método no mejora el desempeño de la clasificación para el problema planteado. Esto podría deberse a los datos de prueba obtenidos, los cuales son considerablemente desequilibrados, dificultando el proceso de clasificación.

3.5.3. *Learning to Rank*

En base a los resultados anteriores se observa que la clasificación es un enfoque más bien ingenuo, ya que no resuelve directamente el problema de seleccionar *keywords* para el propósito que se requiere. Para generar una *tag cloud* se necesita más bien asignar un orden y seleccionar las mejores *keywords*, y tener las potenciales *keywords* clasificadas en n clases no permite eso por sí solo. Es por esto que se acude a técnicas de *learning to rank* para entrenar un modelo que genere un ordenamiento para las *keywords* descritas en la sección 2.4. Además, se quiere comparar el desempeño de las técnicas en esta tarea, dado que existen tres enfoques distintos. Se utilizan las siguientes:

1. Regresión lineal, como enfoque por puntos, utilizando la librería Scikit-learn.
2. Ranking SVM, como enfoque por pares, para el cual se utiliza la implementación en

PySofia[17], un *wrapper* para Python de la librería de C++ sofia-ml[18].

3. LambdaMART, como un segundo enfoque por pares, para el cual se utiliza Pyltr[19]
4. AdaRank, como enfoque por listas[20].

Cada una de estas implementaciones genera un ordenamiento distinto, asignando a cada *keyword* un puntaje. Los puntajes asignados por cada modelo varían incluso en orden de magnitud, siendo algunos valores decimales y otros valores entre 100 y 400. Sin embargo, lo relevante de los resultados no son los valores, sino el orden relativo entre estos.

3.6. Creación de *tag clouds*

Finalmente, una vez seleccionadas las *keywords*, se realiza la creación de la visualización final, creando una para cada técnica de *learning to rank* utilizada. Se utiliza la librería Wordcloud [21], la cual permite la creación de las nubes según la frecuencia de cada palabra, modificar los colores y orientación de las palabras, definir los tamaños mínimos y máximos, entre otras cosas.

Para la creación de las *tag clouds* en primer lugar, dado que, como se menciona anteriormente, los puntajes asignados por las distintas técnicas de *learning to rank* varían en su rango, se estandarizan estos valores, asignando un valor nuevo a cada *keyword* según su posición en el ordenamiento. Este valor se utiliza como la "frecuencia" de la *keyword*, con la cual se define el tamaño de la letra utilizada para esta. Cabe notar que el tamaño de la letra usado en la *keyword* con mayor puntaje es variable, ya que depende del tamaño disponible. Esto se traduce en que si se trata de una muy larga o con muchas palabras, el tamaño de la letra sería más pequeño que en el caso de una más corta. A su vez, esto afecta el tamaño de todas las *keywords* de menor puntaje, ya que cada una tiene que ser más pequeña que la anterior.

3.7. Sistema final

Como se plantea inicialmente, parte del objetivo es que la herramienta creada se encuentre disponible en un sistema abierto para cualquier usuario. Es por esto que se crea un sitio web sencillo que permita la creación de *tag clouds* a partir del link a un perfil de DBLP. El desarrollo del sitio se divide en dos partes: un servicio encargado de generar las *keywords* y el almacenamiento de datos, el cual provee una API[22] para acceder a estos datos, y por otra parte una aplicación web que hace uso de esta API y utiliza los datos obtenidos de esta para generar y mostrar las *tag clouds*. A continuación se describen ambos procesos.

[17] https://github.com/rth/pysofia
[18] https://code.google.com/archive/p/sofia-ml/
[19] https://github.com/jma127/pyltr
[20] https://github.com/rueycheng/AdaRank
[21] http://amueller.github.io/word_cloud/
[22] *Application programming interface* o interfaz de programación de aplicaciones.

3.7.1. API

Se desarrolla una API creada con Flask[23], microframework para la creación de aplicaciones web en Python, y PostgreSQL[24] para manejar la base de datos, mediante SQLAlchemy, herramienta de manipulación de base de datos para Python. Esta actúa como intermediario entre la interfaz con la cual interactúa el cliente y el servidor, y es el medio a través del cual se envían los datos a ambas partes. La API habilita un solo punto de acceso, `/api/keywords`, el cual, dado el URL de un perfil de DBLP, devuelve las *keywords* seleccionadas y su ordenamiento realizado por cada uno de los modelos de *learning to rank*, en formato JSON.

Las *keywords* seleccionadas se obtienen de dos formas. En primer lugar, si las *keywords* nunca han sido generadas, se utiliza la herramienta creada para realizar todo el proceso: la obtención de las publicaciones, la extracción del texto de estas, la extracción de las *keywords* candidatas y la selección y re-ordenamiento de estas mediante los modelos de *learning to rank*. Dado que este proceso es demoroso, se genera un caché de estos resultados, los cuales son almacenados en la base de datos. Así, la segunda forma de obtener las *keywords* es en el caso de que ya se encuentre el resultado en el caché, este se extrae de la base de datos, resultando en un proceso mucho más rápido.

El modelo de datos tiene una estructura bastante simple, consistiendo de dos entidades, "Author" y "Result". En la figura 3.5 se puede ver un diagrama de las entidades y como se relacionan. La entidad "Author" almacena los nombres de los autores y su identificador, para lo cual se utiliza el URL a su perfil de DBLP. Por su parte, la entidad "Result" almacena los resultados de una selección de *keywords* para cierto autor y la fecha y hora en la que fueron generados. Cabe notar que un autor puede tener varios resultados, ya que estos podrían quedar obsoletos en el caso de que un autor tenga varias publicaciones nuevas que no fueron consideradas al momento de generar estos resultados. Es por esto que no se considera los resultados guardados que sean muy antiguos, en cuyo caso se generan nuevamente.

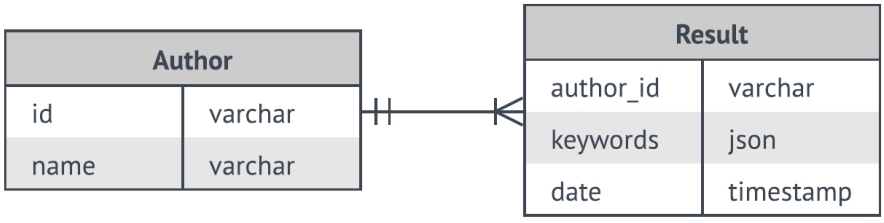

Figura 3.5: Modelo de datos.

[23] http://flask.pocoo.org/
[24] https://www.postgresql.org/

3.7.2. Interfaz de usuario

Por otra parte, se tiene una aplicación web, creada con ReactJS[25], con la cual interactúa directamente el usuario. La aplicación consiste de dos principales vistas. La primera vista, la cual se puede ver en la figura 3.6, permite ingresar el URL de un perfil de DBLP y generar las *tag clouds*, las cuales se generan con un componente de ReactJS[26] construido con d3-cloud[27]. Las distintas figuras creadas se pueden navegar mediante botones en la parte inferior, que permiten cambiar la visualización que se está observando por la siguiente. Esto se realiza mediante una solicitud a la API, de la cual se obtiene un JSON con los resultados como respuesta, los cuales se utilizan para poblar las *tag clouds*. La segunda vista permite la edición de una *tag cloud*, en particular agregar o quitar *keywords*, y descargarla como imagen. En la figura 3.7 se puede observar esta vista.

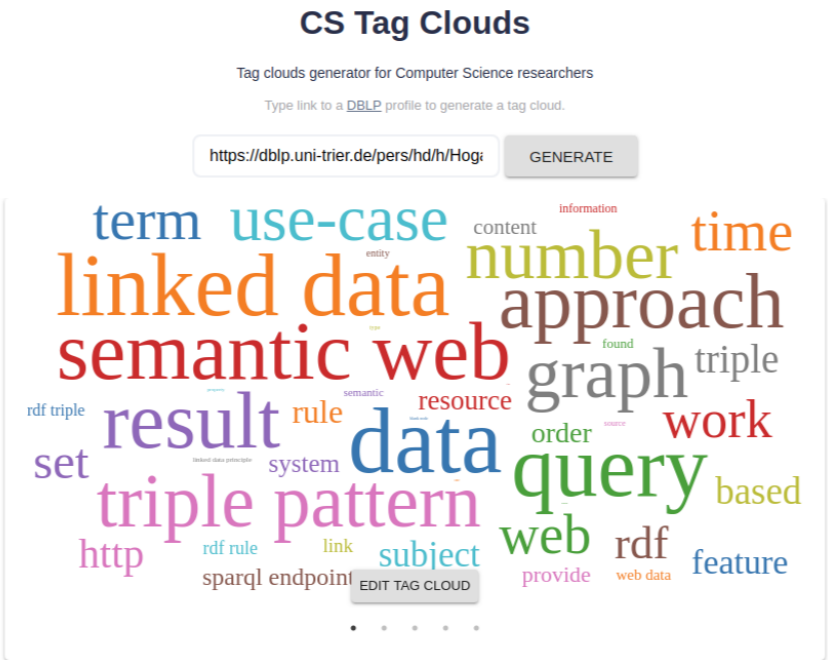

Figura 3.6: Vista inicial luego de la generación de las *tag clouds*.

[25] https://reactjs.org/
[26] https://github.com/Yoctol/react-d3-cloud
[27] https://github.com/jasondavies/d3-cloud

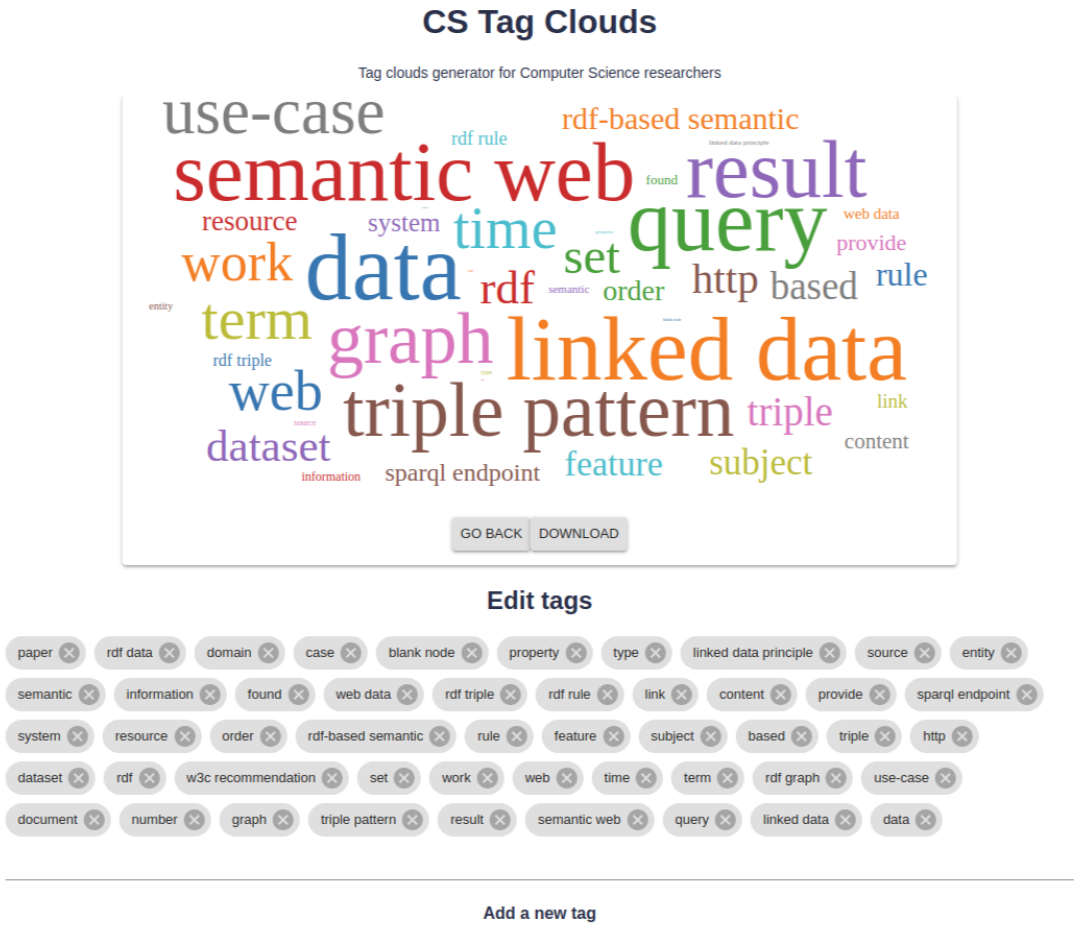

Figura 3.7: Vista de edición.

Capítulo 4

Evaluación de la solución

En este capítulo se muestran en detalle las evaluaciones realizadas a la herramienta creada. Las pruebas se enfocan principalmente a evaluar los métodos de *learning to rank* y si estos efectivamente benefician el proceso de selección de las *keywords*. Por otra parte, también es de interés la evaluación del proceso de extracción de las publicaciones, cuánto tiempo toman y cuántas son extraídas, ya que esto influencia la calidad de las *keywords* extraídas. Para esto se plantean las siguientes preguntas, las cuales son respondidas a lo largo del capítulo:

- ¿Cuántas publicaciones se pueden extraer?
- ¿Cuánto toma el proceso de extracción de las publicaciones?
- ¿Los modelos de *learning to rank* benefician el proceso de extracción? En particular, ¿son mejores que un ordenamiento aleatorio o que el ordenamiento según el puntaje RAKE?
- ¿Qué tan similares son los modelos entre sí?
- ¿Qué tan buena es la calidad de las *tag clouds* finales?

4.1. Conjunto de datos

A continuación se describe el conjunto de datos utilizado para la evaluación. Además, se describe su proceso de obtención, desde la obtención de las publicaciones y la extracción de las *keywords*, hasta la construcción de un "*gold standard*" a partir de estas.

4.1.1. Extracción de *keywords*

Como primer paso para la obtención del conjunto de datos se realiza la extracción de publicaciones descrita en la sección 3.3. Cabe notar que en una primera instancia solo se extraen los PDFs encontrados en páginas estáticas. Para esto se utilizan las publicaciones de 21 profesores de jornada completa del DCC, principalmente debido a la mayor facilidad para consultar su opinión y que se tiene una noción del área de investigación de cada uno.

En total se extraen 1091 papers, con una recuperación promedio de 0.58. El tiempo de extracción es 0:13:19 en promedio por autor, pero demora como máximo 40 minutos. Esta es la etapa más lenta del proceso completo, tomando en total cerca de 4.5 horas en extraer todas las publicaciones. Una vez finalizada la obtención de las publicaciones, se realiza la extracción del texto de estas, proceso que toma alrededor de 9 minutos, con un total de 3 horas en la extracción de todos los textos. Luego de esto, para la extracción de las *keywords* se utiliza el algoritmo RAKE de la forma descrita en la sección 3.4.3. Este proceso toma entre 3 y 4 minutos por persona.

Tabla 4.1: Resumen de recuperación y tiempos del proceso de extracción de *keywords*.

	Promedio	**Mediana**	**Mínimo**	**Máximo**
Recuperación por autor	0.581	0.586	0.274	0.881
Tiempo extracción PDFs por paper	17s	14s	8s	53s
Tiempo conversión por paper	11s	10s	4s	33s
Tiempo extracción keywords por paper	6s	5s	2s	20s
Tiempo total por paper	34s	29s	18s	73s

El proceso completo toma alrededor de 25 minutos por persona, con un tiempo de 9 horas en total. En la tabla 4.1 se muestra un resumen de la recuperación lograda y los tiempos de cada proceso de la extracción de *keywords* por publicación.

4.1.2. Etiquetado de las *keywords*

A partir del conjunto de *keywords* extraído, se seleccionan las 100 mejores, según el puntaje obtenido con RAKE, por persona. Se invita a los profesores a participar en una evaluación inicial, la cual consiste en etiquetar las *keywords* extraídas de sus publicaciones con los siguientes puntajes:

- 1 - no tiene nada que ver, no debería estar en la lista.
- 2 - no es muy relevante, sería mejor que no esté en la lista.
- 3 - es relevante, podría estar en la lista.
- 4 - es importante, está bien que esté en la lista.
- 5 - es una *keyword* muy importante, tiene que estar en esta lista.

Definiendo de esta forma qué *keywords* son mejores que otras. Además, se les da la posibilidad de agregar *keywords* que consideren necesarias pero no se encontraban en el top 100.

De esta evaluación inicial se obtienen 12 respuestas con 1126 elementos en total, cuya distribución de puntajes se puede observar en la figura 4.1. De la figura podemos observar

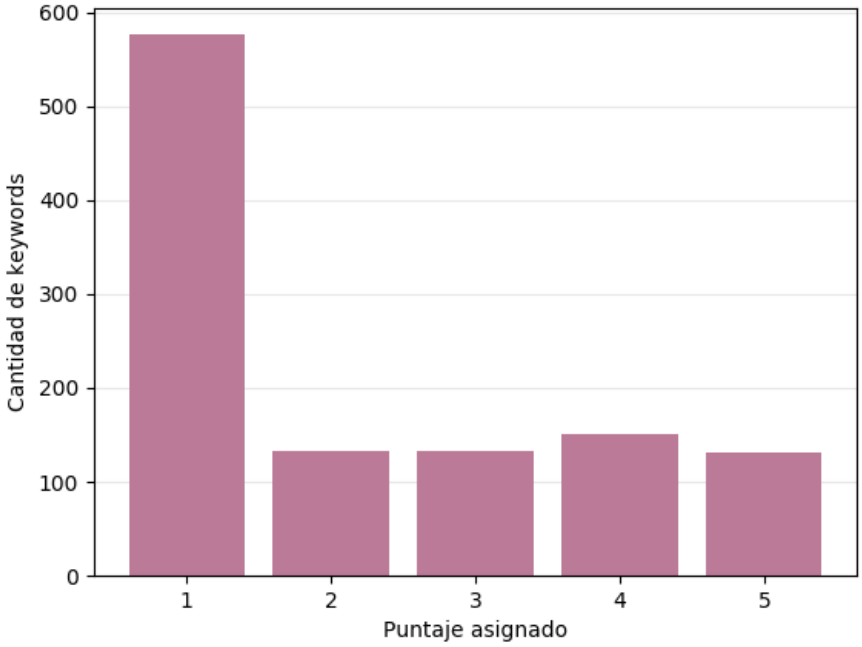

Figura 4.1: Distribución de puntajes

que la cantidad de elementos a los cuales se le asigna el puntaje mínimo es considerablemente mayor a los demás, siendo 576 en esta categoría y 550 en las otras cuatro combinadas. Las *keywords* a las cuales se le asigna el puntaje mínimo en general corresponden a palabras comunes en el contexto de Ciencias de la Computación.

4.1.3. Características

Luego de esto, se extraen las características definidas para cada uno de los elementos etiquetados. De las características se percibe lo siguiente:

- La mayoría de las *keywords* del conjunto de datos tiene un tf-idf bajo, entre 0 y 0.1.
- La mayoría también se encuentra en Wikipedia.
- En general son *keywords* de una sola palabra.

En la figura 4.2 se pueden observar los histogramas de todas las características.

En un comienzo se proponen nueve características, en la sección 3.5.1, sin embargo, no se utilizan todas en el entrenamiento final de los modelos. Una de las características descartadas es la profundidad de frase, la cual se observa que acumula valores muy altos para palabras comunes como "*problem*", "*set*" y "*number*", y se cree que esto podría confundir a los modelos. Por otra parte, se cree que el año de la primera aparición no es muy definitivo a la hora de decidir si una *keyword* es relevante o no, en particular dado que se tiene la diferencia entre el año más reciente y el primer año de aparición como una característica aparte. Esta

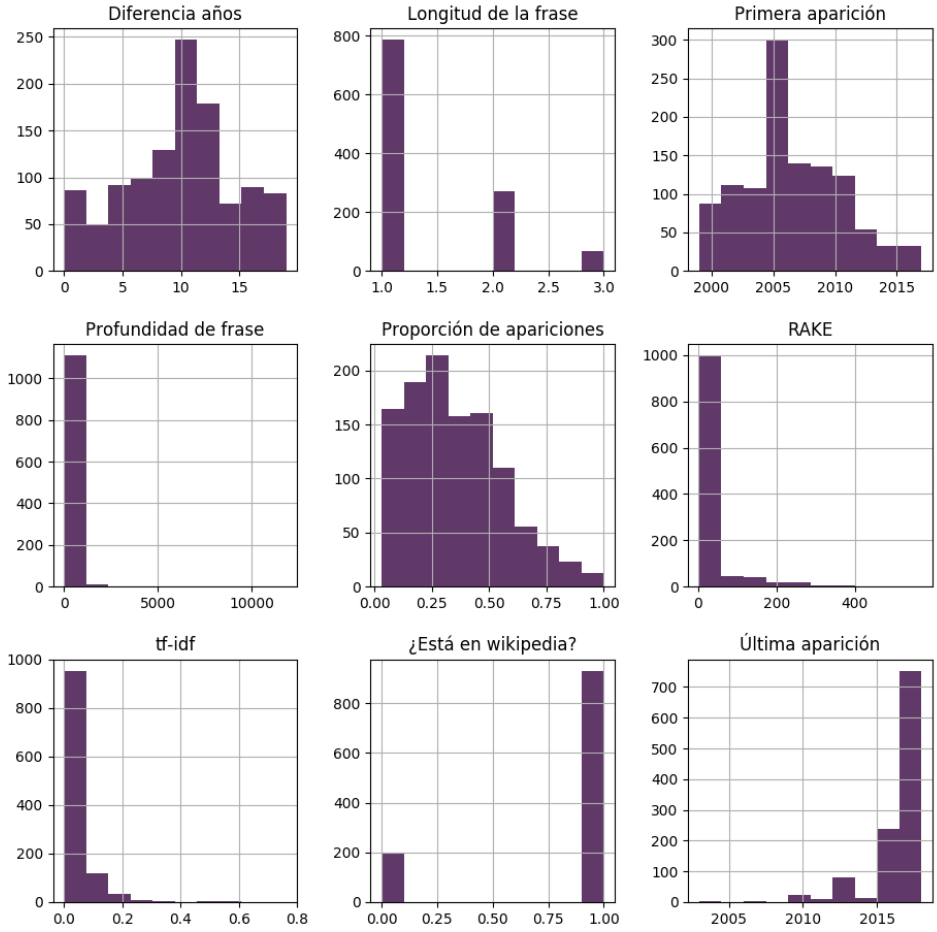

Figura 4.2: Histograma de características.

información pareciera ser de más utilidad de esta forma más explícita para los modelos.

Finalmente el conjunto de datos corresponde a las *keywords* etiquetadas y con sus características respectivas. En particular el conjunto se considera como 12 ordenamientos ideales, dados por los puntajes asignados de las 12 listas de *keywords* etiquetadas, al cual se referirá como "*gold standard*".

4.2. Evaluación de los modelos de *learning to rank*

Para la evaluación de los modelos de *learning to rank* escogidos, los cuales se mencionan en la sección 3.5.3, se utilizan las siguientes implementaciones:

1. Regresión lineal.
2. Ranking SVM, con una configuración de:
    - Máximo número de iteraciones: 1000
3. LambdaMART, con una configuración de:
    - Métrica: ganancia acumulada descontada normalizada (NDCG)
    - Número de estimadores: 2000
    - Tasa de aprendizaje: 0.03
4. AdaRank, con una configuración de:
    - Máximo número de iteraciones: 100
    - Mínimo número de iteraciones para parada temprana: 30
    - Métrica: ganancia acumulada descontada normalizada (NDCG)

Estos modelos se compraran con otros dos ordenamientos:

1. Ordenamiento según el puntaje obtenido por RAKE, ya que este es el método base utilizado para la extracción de *keywords* candidatas, por lo cual es de interés ver si los modelos de *learning to rank* presentan una mejora en comparación.
2. Ordenamiento aleatorio.

Los parámetros e hiperparámetros utilizados se definen partiendo con la configuración por defecto y modificando algunos valores. Con cada configuración se observan los resultados de NDCG en una validación cruzada, para decidir si estos mejoran o empeoran con respecto a la configuración original. Por otra parte, también se reserva un pequeño conjunto de datos de prueba, para evaluar nuevamente el desempeño del modelo con la configuración escogida.

Los datos utilizados para la validación cruzada corresponden a las 12 listas de *keywords* etiquetadas referidas anteriormente. Para la validación final se utilizan las mejores 500 *keywords* candidatas obtenidas mediante RAKE por cada autor, a partir de las publicaciones extraídas.

4.2.1. Validación cruzada

En primer lugar, para evaluar el desempeño de cada modelo, se realiza una validación cruzada. Para esto se divide el conjunto de datos en 12 grupos, correspondientes a las 12 listas de *keywords*. De esta forma, para cada grupo:

1. Se entrenan los modelos con el resto de los grupos.
2. Se evalúan los modelos con el grupo seleccionado.

Para la evaluación de los modelos se utilizan las métricas descritas en la sección 2.4.1. Los resultados se presentan a continuación.

Ganancia acumulada descontada normalizada (NDCG)

Como se menciona anteriormente, la ganancia acumulada descontada (DCG) es una métrica de evaluación de rendimiento para modelos de ordenamiento, la cual se utiliza para comparar entre el ordenamiento de una lista de elementos generado por cada modelo y el orden ya conocido del *gold standard*. En particular se utiliza la versión normalizada (NDCG), para comparar listas distintas, y la ganancia acumulada descontada normalizada para los top 5 elementos (NDCG@5). Esta última es interesante, ya que en el caso de una *tag cloud* las *keywords* más visibles corresponden a las top 3 o 5. En las figuras 4.3 y 4.4 se pueden observar los resultados obtenidos de NDCG y NDCG@5 respectivamente para cada modelo, comparados con el ordenamiento inicial generado por el solo uso de la métrica RAKE y un ordenamiento aleatorio.

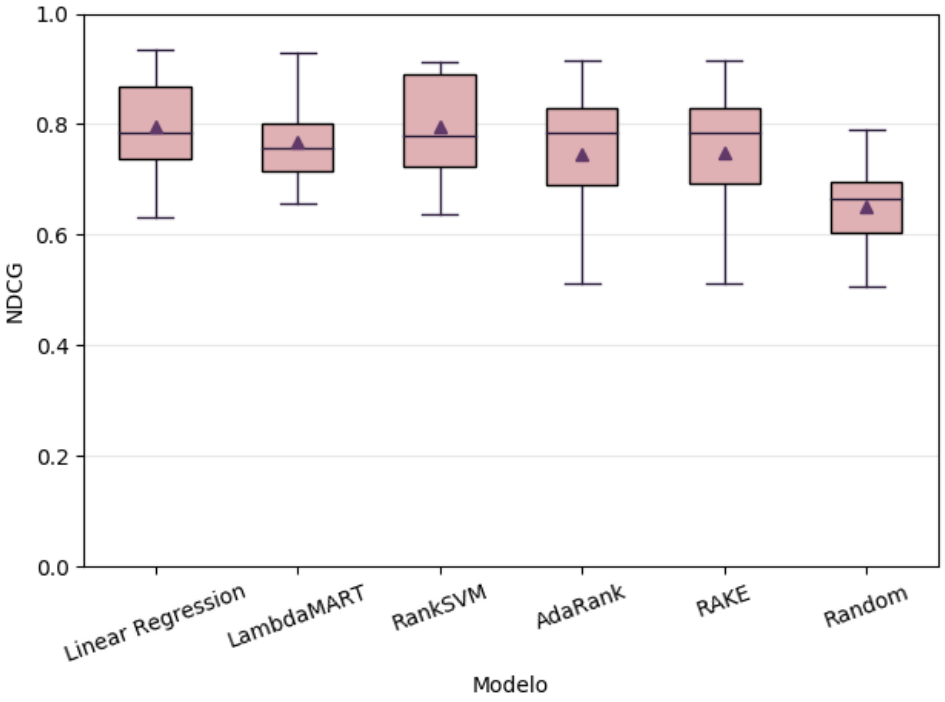

Figura 4.3: NDCG

Lo primero que se observa es que todos los métodos son al menos mejores que un ordenamiento aleatorio en promedio, los cuales se encuentran indicados por un triángulo en ambas figuras. Se observa además que todos los métodos tienen valores de NDCG bastante similares, de alrededor de 0.8, por lo cual ninguno destaca por sobre los demás según esta métrica. Si tomamos el NDCG@5, se tiene una situación similar, pero con valores inferiores, de alrededor de 0.5. Con ambas métricas es difícil comparar a los métodos entre sí.

Entre las diferencias notamos que LambdaMART tiene una menor varianza, pero su desempeño promedio es inferior. Por otra parte RankSVM y regresión lineal, a pesar de ser modelos más simples, tienen los mejores promedios, pero una varianza superior a la de LambdaMART. Finalmente, se observa que el desempeño de AdaRank y RAKE es extremadamente parecido,

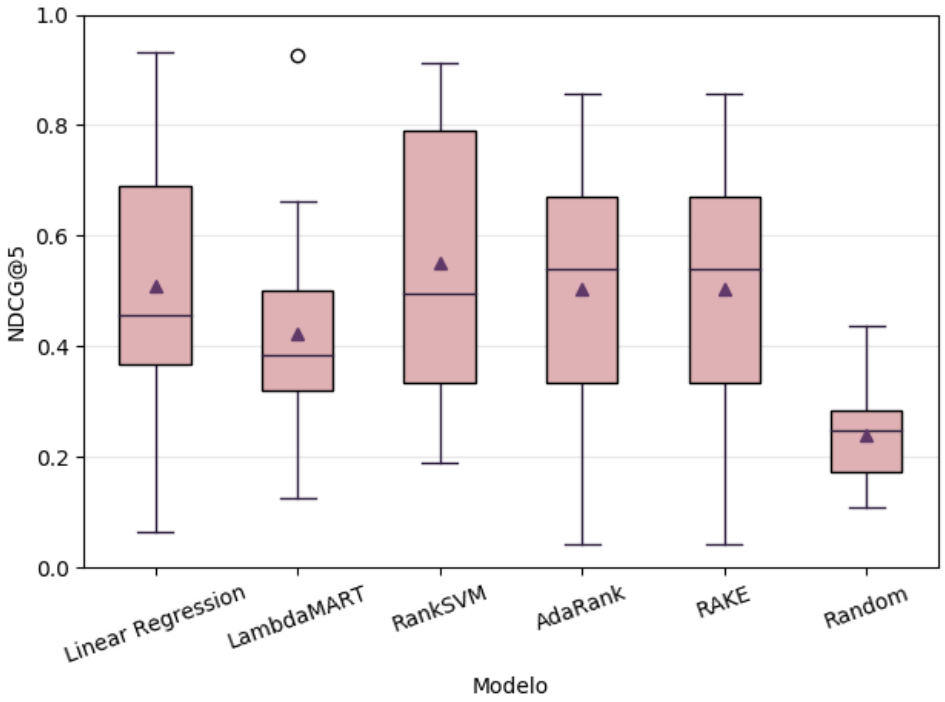

Figura 4.4: NDCG@5

con diferencias muy pequeñas, lo cual sugiere que en el entrenamiento AdaRank se basa demasiado en RAKE. En AdaRank y RAKE los resultados son bastante inconsistentes, obteniendo buenos resultados en algunos casos y resultados muy malos en otros.

Tau de Kendall

El coeficiente tau de Kendall permite calcular la correlación entre dos listas ordenadas. En particular, se utiliza el coeficiente tau-b, descrito anteriormente, el cual realiza un procedimiento distinto en caso de encontrar empates, ya que en el ordenamiento ideal se presentan varios empates. Esto permite realizar una comparación de la correlación entre todos los métodos utilizados, además de un ordenamiento aleatorio. Esta información se puede observar en resumen en la figura 4.5, la cual, dado dos tipos de ordenamientos, muestra el promedio de los índices de correlación obtenidos por cada grupo de la validación cruzada. En la tabla A.1 del apéndice A se pueden observar todos los valores del índice de correlación de cada ordenamiento con respecto al *gold standard*.

Como es de esperar, se observa que la correlación entre el ordenamiento aleatorio y el *gold standard* es cercana a 0. Además, con la comparación de correlación se evidencia aún más la similitud entre el ordenamiento de AdaRank y el de RAKE, los cuales tienen un $\tau = 1$, lo cual indica una correlación perfecta. Esto podría indicar, como se sospecha anteriormente, que AdaRank se basa demasiado en la característica RAKE. Ambos tienen un índice de correlación bastante bajo con respecto al *gold standard*, cercano a 0. Por otra parte, los otros tres modelos, LambdaMART, RankSVM y regresión lineal, son bastante similares entre sí.

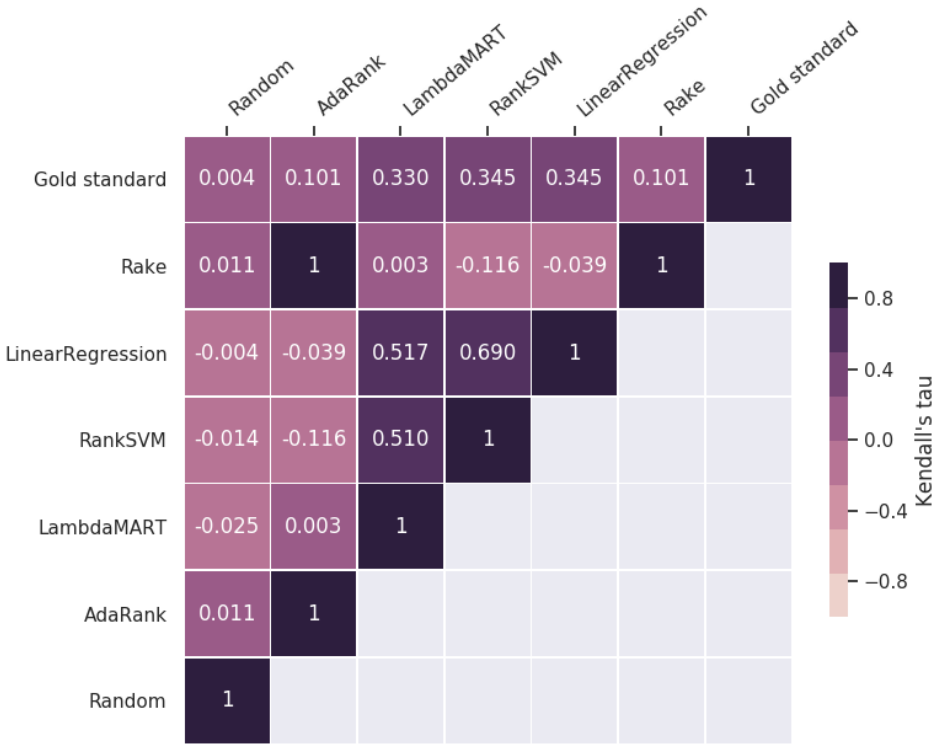

Figura 4.5: Mapa de calor de la correlación entre los modelos.

Los tres tienen un índice de correlación similar con respecto al *gold standard*, de alrededor de 0.35, el cual es considerablemente mayor que el de los dos métodos anteriores.

Valores p

Finalmente, para descartar que los resultados anteriores sean solo casualidad, se calcula el valor p para un test de hipótesis cuya hipótesis nula es la ausencia de correlación entre cada modelo de ordenamiento y el *gold standard*, es decir, un valor de $\tau = 0$. El valor p, en este caso, se entiende como la probabilidad de que los métodos de ordenamiento sean efectivos bajo la hipótesis nula. De esta forma, si el valor p es menor que cierto nivel de significancia (típicamente 0.05) entonces la hipótesis nula será rechazada. En el caso de que el valor p sea muy grande no es posible rechazar la hipótesis nula. En la tabla 4.2 se puede ver los valores p obtenidos por cada método.

En primer lugar, se puede observar que en promedio LambdaMART, RankSVM y regresión lineal tienen valores p suficientemente pequeños, siendo los valores de regresión lineal los más bajos de los tres en promedio. Aún así, LambdaMART y RankSVM contienen valores particulares que son altos. LambdaMART en general tiene valores bastante bajos con la excepción de un caso en particular que tiene un valor p de 0.5. Cabe notar que para este mismo caso los otros modelos tienen valores bastante bajos. Los resultados de RankSVM

Tabla 4.2: Tabla completa de valores p.

Autores	Lambda MART	AdaRank	RankSVM	Linear Regression	Rake
A	0.01428361	0.04521409	0.06184521	0.04302632	0.04521409
B	5.60E-10	0.77779265	3.89E-08	3.89E-08	0.77779265
C	0.52601113	0.00732027	5.84E-09	0.00133386	0.00732027
D	0.00004302	0.03978224	0.00001268	0.00000097	0.03978224
E	0.00000386	0.62843507	0.00018180	0.00001369	0.62843507
F	0.00000040	0.00050270	6.32E-09	9.32E-08	0.00050270
G	0.00184394	0.74173164	0.00865524	0.00056594	0.74173164
H	0.00069646	0.96828438	0.00006843	0.00035328	0.96828438
I	0.00000019	0.00106845	0.00001479	0.00000242	0.00106845
J	0.00000037	0.98353051	0.00021982	0.00000151	0.98353051
K	0.00001720	0.75699485	0.00004910	0.00000114	0.75699485
L	4.47E-09	0.36641803	5.34E-11	1.01E-10	0.36641803
Promedio:	0.0452	0.4431	0.0059	0.0038	0.4431

presentan una situación similar, aunque en este caso el valor máximo es 0.06.

Nuevamente AdaRank y RAKE tienen un peor desempeño que los otros métodos, con valores p muy altos, los cuales son incluso comparables con los valores de un ordenamiento aleatorio. Esto indica que no es posible rechazar la hipótesis nula, por lo cual probablemente no existe una correlación entre los ordenamientos obtenidos mediante estos métodos y el ordenamiento ideal.

4.2.2. Validación final

Finalmente, se invita nuevamente a los profesores a participar en una encuesta de evaluación final. Esta consiste de asignar una calificación de 1 a 5, siendo 1 muy mala y 5 muy buena, a seis *tag clouds* generadas con los modelos de *learning to rank*, RAKE y una aleatoria. Para no influenciar las opiniones por temas estéticos, las *tag clouds* generadas se mantienen lo más neutral posible, utilizando un color neutral y una orientación horizontal para todas las palabras. Cabe mencionar que en el caso de los profesores que ya habían dado su opinión en la encuesta inicial, no se ocupan los datos obtenidos de sus respuestas individuales anteriores para entrenar los modelos utilizados al generar sus *tag clouds*. Por otra parte, para los demás profesores se ocupan todos los datos de entrenamiento. En la figura 4.6 se puede ver la distribución de los puntajes por cada método.

Los resultados son bastante variados. No se observa una preferencia absoluta por un método por sobre los demás, sino que el preferido varía según cada persona. Una observación interesante es que los resultados no concuerdan completamente con los resultados obtenidos de las métricas anteriores. Un ejemplo es el caso de la regresión lineal, la cual, a pesar de ser el modelo más simple, presenta en general un valor de NDCG bastante alto y una correlación con el *gold standard* más alta que los demás según el índice de correlación de Kendall.

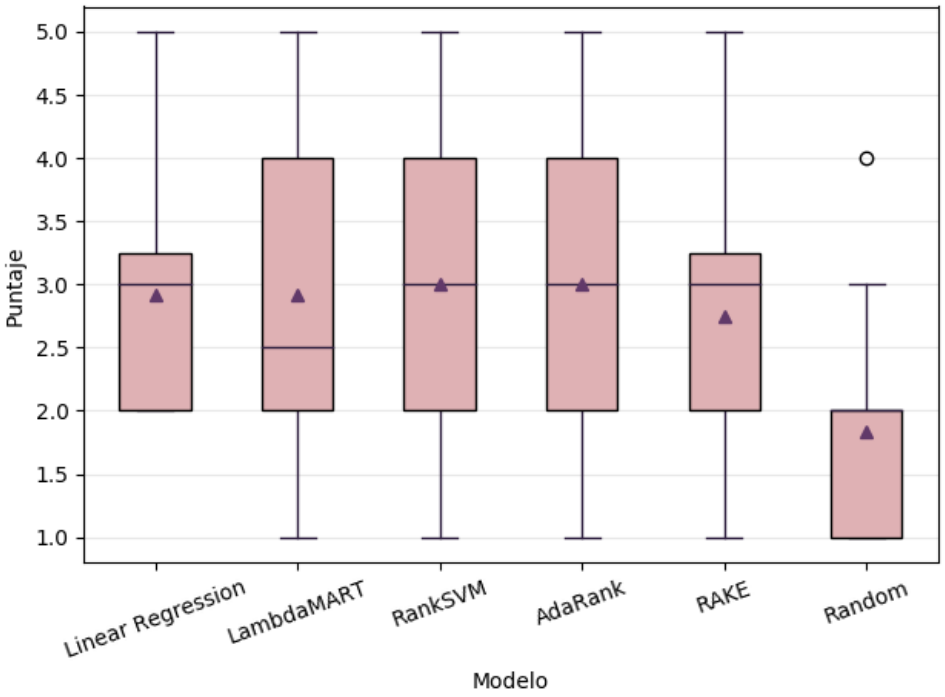

Figura 4.6: Distribución de puntajes a cada método de selección de *keywords*

Sin embargo, sus puntajes son levemente inferiores. Por otra parte, AdaRank y RAKE, que tienen un índice de correlación muy bajo con respecto al *gold standard*, logran puntajes altos en algunos casos.

4.3. Discusión

En este capítulo se muestran los resultados de las evaluaciones realizadas a la herramienta creada. Para esto se definen previamente un conjunto de preguntas a responder. A continuación se analizan los resultados obtenidos y cómo estos responden las preguntas planteadas.

Extracción de publicaciones

En primer lugar, se describe el conjunto de datos utilizado para las evaluaciones. Del proceso de obtención de este conjunto, se observa que la extracción de publicaciones toma alrededor de 13 minutos en promedio por persona y 4.5 horas en total, siendo la etapa más demorosa del proceso. A este lo sigue la extracción del texto de las publicaciones, con un tiempo promedio de 8 minutos por persona y un total de 3 horas. En total se extraen 1091 papers, con una recuperación promedio de 0.58 por persona. Dado que, como se menciona en la sección 4.1, en esta instancia solo se extraen las publicaciones encontradas en páginas estáticas y que existen publicaciones a las cuales no se puede acceder mediante la suscripción de la universidad, esta recuperación promedio es esperable. Con la incorporación de la

metodología para extraer las publicaciones encontradas en páginas dinámicas, descrita en la sección 3.3, la recuperación promedio aumenta a 0.62 por persona, y el mínimo pasa de 0.274 a 0.337.

Desempeño de los modelos con respecto a RAKE y orden aleatorio

Se realiza una validación cruzada de las métricas de desempeño, NDCG y el índice correlación de Kendall. Los valores de NDCG son bastante similares entre sí, de alrededor de 0.8. De todos modos, la regresión lineal y RankSVM presentan valores levemente superiores. Los valores obtenidos son bastante buenos, sin embargo, los valores de NDCG@5 son menores, de alrededor de 0.5, pero muy variables. Esto indica que el ordenamiento de las primeras cinco *keywords* de los modelos no se asemeja tanto al *gold standard*, aunque existen excepciones, pero el ordenamiento completo si. Aún así, todos los modelos tienen un desempeño mejor al del ordenamiento aleatorio, pero no presentan valores considerablemente superiores a los de RAKE. De los valores del índice de correlación de Kendall con respecto al *gold standard* se observa que LambdaMART, RankSVM y regresión lineal tienen valores considerablemente mayores en contraste con RAKE y el ordenamiento aleatorio. De esto se concluye estos tres modelos anteriores generan ordenamientos bastante más similares al ideal que Rake y AdaRank.

Similitud entre los modelos

De los otros valores del índice de correlación se observa que AdaRank es extremadamente similar a RAKE, lo cual sugiere que se basa demasiado en esta característica para su entrenamiento. Los otros tres modelos, LambdaMART, RankSVM y regresión lineal, tienen un índice de correlación relativamente alto entre sí, lo cual sugiere que existe cierta similitud entre ellos. Cada uno de estos, además, tiene un índice de correlación bajo con respecto a AdaRank y RAKE. Cabe notar que incluso en el caso de los modelos con una correlación más alta entre sí, dado que difieren levemente, estas diferencias podrían hacerse muy notorias en la visualización final, en particular si son las primeras cinco *keywords* las que son distintas.

Calidad de las *tag clouds* finales

La validación final, orientada a evaluar la calidad de las *tag clouds*, indica que no habría una preferencia absoluta por un método por sobre los demás, sino que el preferido varía según cada persona. Las únicas *tag clouds* a las cuales se le asignan solo puntajes bajos es la creada de forma aleatoria. Esto discrepa un poco con algunos de los resultados anteriores, según los cuales la regresión lineal era generalmente el modelo con mejor desempeño, y AdaRank por su parte era uno de los peores. Esto podría deberse a la naturaleza de las *tag clouds*, ya que las palabras con un tamaño de letra más grande destacan demasiado por sobre las demás, en particular las 3 a 5 más grandes. De esta forma, un método capaz de escoger solo las mejores cinco palabras de forma correcta probablemente daría una mejor impresión que uno que logra un orden más similar al ideal, pero las cinco mejores palabras no son tan buenas.

En este caso, los valores de NDGC@5, en la figura 4.4, podrían ser más útiles para predecir el desempeño de los modelos.

Aún así, en la mayoría de los casos se le asigna el puntaje máximo a al menos una de las *tag clouds* generadas. De esto se concluye que aunque no exista un solo modelo que se ajuste completamente a todos los casos, existe al menos uno de estos que se aproxima. Por otro lado, se observa que en general el tipo de *keywords* seleccionadas por cada modelo son bastante distintas, dando énfasis a aspectos distintos, por lo que acá entran en juego las preferencias de cada uno. Esto se percibe incluso desde la primera encuesta, en la cual se observa que algunos encuestados prefieren términos más técnicos y otros términos más genéricos. Un ejemplo de esto es el caso de la *keyword* "*computer science*", que debería aplicar a todos los encuestados, pero de forma muy genérica, y a la cual algunos le asignaron el puntaje máximo y otros el puntaje mínimo.

Otra observación llamativa es que en algunos casos los profesores prefieren que las *keywords* reflejen su trabajo más reciente. Esto en particular ocurre en casos en los que el tema de investigación del académico fue evolucionando a través de los años y las *keywords* seleccionadas de publicaciones muy antiguas ya no son relevantes en su trabajo actual. Esto se toma en consideración desde un comienzo, debido a comentarios de algunos profesores, por lo cual se incluye como característica el año de la mención más reciente de la *keyword*. Sin embargo, se cree que esta no toma tanto protagonismo, posiblemente debido a que para otros profesores esto no era un problema, ya sea porque se han mantenido constantes en su tema de investigación o porque no les molesta que aparezcan *keywords* más bien históricas, y sus respuestas en la encuesta inicial reflejan esto. Una de las soluciones que se plantea a esta situación es la de permitir filtros en el sistema final que permitan definir entre qué años de publicación deberían estar los papers utilizados en la extracción de las *keywords*.

Dada la subjetividad involucrada, es difícil lograr que un modelo de ordenamiento sea el mejor universalmente para este uso. Sin embargo, como se menciona anteriormente, es posible lograr que al menos uno se aproxime. Es por esto que en el sistema final se incluyen todas las variantes de *tag clouds* creadas, para que cada usuario pueda elegir la que se acerque más a sus preferencias. Por otra parte, también se considera de importancia dar la posibilidad al usuario de editar la *tag cloud* final para que esta se acerque aún más a sus preferencias, permitiendo eliminar *keywords* que no le parezcan adecuadas o agregar alguna que consideren faltante.

Capítulo 5

Conclusiones y trabajo a futuro

En este trabajo de memoria se busca la creación de una herramienta de uso abierto que permita la creación de *tag clouds* para perfiles de DBLP. Esto se realiza con el objetivo de facilitar la información resumida de los tópicos de investigación de académicos de Ciencias de la Computación a cualquiera que la requiera, o incluso ser integrada en otros sistemas. Para esto, el trabajo se enfoca en la investigación de modelos de *learning to rank* y la comparación de su desempeño en la tarea de definir las *keywords* más relevantes para un investigador de Ciencias de la Computación.

La solución propuesta cumple con los objetivos planteados inicialmente, permitiendo, dado un perfil de DBLP, la obtención de las publicaciones encontradas en el perfil, la extracción de potenciales *keywords* y la selección de las *keywords* más relevantes según cuatro modelos de ordenamiento. Según estos modelos se crea una variante de *tag cloud* por cada uno. Finalmente, se cuenta con un sitio web que permite el uso de la herramienta para cualquier usuario.

Se realiza una evaluación, tanto a los ordenamientos en si, con una validación cruzada de las métricas de desempeño, NDCG y el índice correlación de Kendall, como a las visualizaciones finales, mediante una validación final con una encuesta a profesores del DCC. En el primer caso, los resultados indican que RankSVM y regresión lineal son los dos modelos con mejor desempeño, seguidos por LambdaMART. Sin embargo, la validación final indica que no habría una preferencia absoluta por un método por sobre los demás, sino que varía según cada persona, pero en la mayoría de los casos se le asigna el puntaje máximo a al menos una de las *tag clouds* generadas. Esta discrepancia podría deberse a la naturaleza de las *tag clouds*, ya que las palabras con un tamaño de letra más grande destacan por sobre las demás, en particular las 3 a 5 más grandes. Por otra parte, también existe un elemento de subjetividad en juego, el cual podría afectar la preferencia de los usuarios.

Por supuesto, es posible continuar el trabajo, desde mejorar el desempeño de los modelos de ordenamiento, hasta implementar nuevas funcionalidades para el sitio web. Uno de los aspectos más interesantes es intentar mejorar el desempeño de los modelos mediante la obtención de más datos que permitan un mejor entrenamiento. Una de las ideas es utilizar el sitio web mismo, dado que se permite la modificación de las *keywords* incluidas en las *tag*

clouds generadas, se podría usar esto mismo como retroalimentación para el sistema. Esto, sin embargo, tiene la desventaja de que en un comienzo probablemente no se cuente con muchos datos, ya que esto depende del uso que se le dé a la herramienta. Otra idea es el uso de datos externos, para lo cual se podrían considerar fuentes confiables, como las *keywords* que a veces son incluidas en las mismas publicaciones, o posiblemente menos confiables, como basadas en reseñas de publicaciones encontradas en la web. Sin embargo, la recopilación de estos datos sería más complicada y quizás un desafío de automatizar, por lo cual se requeriría más tiempo.

Un elemento que se quiere mejorar a futuro es el tiempo que demora el proceso completo, en el caso de no contar con los datos ya en caché, el cual puede llegar a tomar más de una hora en algunos casos. Dado que la extracción y descarga de las publicaciones es la etapa más lenta del proceso, el foco debería estar en disminuir estos tiempos, de ser posible. Una de las ideas es probar utilizando menos publicaciones, por ejemplo 10 por autor, y observar si la calidad de las *tag clouds* es comparable a la de las generadas utilizando todas las publicaciones. Por otra parte, actualmente, si una publicación ya fue extraída para crear las *tag clouds* de un autor y luego se quiere crear las *tag clouds* de un co-autor de esta misma publicación, esta se vuelve a descargar, a pesar de que ya se cuenta con ella. Otra forma de disminuir los tiempos sería tomar en consideración estas situaciones y aplicar optimizaciones por este lado.

Otro de los aspectos que se podría analizar es el uso del algoritmo KEA, el cual fue descartado en un comienzo dada la falta de un conjunto de datos de entrenamiento. Dado que en el proceso se obtiene un conjunto de datos que puede ser utilizado con ese fin, sería de interés ver si el uso de KEA mejora la selección de *keywords* inicial.

Por otra parte, dado que la herramienta se enfoca solo en investigadores de Ciencias de la Computación, sería interesante extender su uso a otras disciplinas y, en particular, analizar el desempeño de los modelos de ordenamiento en estas.

Finalmente, dado que el sitio web en sí no fue el foco principal del trabajo, hay varias funcionalidades que se quieren agregar a futuro. En primer lugar, permitir filtros en la extracción de los papers. Un ejemplo de esto es el caso de que se quiera formar una idea solo de las investigaciones más recientes de un académico, para lo cual se podrían utilizar solo las publicaciones en un rango de años definido por el usuario. Esto ayudaría tanto para moldear más las visualizaciones creadas a los deseos del usuario, como para agilizar el proceso y disminuir los tiempos de espera. Otra funcionalidad es permitir la personalización de las *tag clouds*, desde los colores hasta el tipo de letra.

Bibliografía

[1] Bateman, S., C. Gutwin y M. A. Nacenta: *Seeing things in the clouds: the effect of visual features on tag cloud selections*. En *HYPERTEXT 2008, Proceedings of the 19th ACM Conference on Hypertext and Hypermedia, Pittsburgh, PA, USA, June 19-21, 2008*, páginas 193–202, 2008. http://doi.acm.org/10.1145/1379092.1379130.

[2] Burges, C. J. C.: *From RankNet to LambdaRank to LambdaMART: An Overview*. Informe técnico, June 2010. https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/.

[3] Cao, Z., T. Qin, T.-Y. Liu, M.-F Tsai y H. Li: *Learning to rank: from pairwise approach to listwise approach*. En *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, páginas 129–136, 2007. http://doi.acm.org/10.1145/1273496.1273513.

[4] Frank, E. y M. A. Hall: *A Simple Approach to Ordinal Classification*. En *Machine Learning: EMCL 2001, 12th European Conference on Machine Learning, Freiburg, Germany, September 5-7, 2001, Proceedings*, páginas 145–156, 2001. https://doi.org/10.1007/3-540-44795-4_13.

[5] Graf, A. B. A. y S. Borer: *Normalization in support vector machines*. En *in Proc. DAGM 2001 Pattern Recognition*, páginas 277–282. SpringerVerlag, 2001.

[6] Halvey, M. y M. T. Keane: *An assessment of tag presentation techniques*. En *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, páginas 1313–1314, 2007. http://doi.acm.org/10.1145/1242572.1242826.

[7] Hsu, C.-W., C. C. Chang y C.-J. Lin: *A Practical Guide to Support Vector Classification*. 101:1396–1400, 2003.

[8] James, G., D. Witten, T. Hastie y R. Tibshirani: *An Introduction to Statistical Learning*. Enero 2013, ISBN 1461471389.

[9] Kuo, B. Y.-L., T. Hentrich, B. M. Good y M. D. Wilkinson: *Tag clouds for summarizing web search results*. En *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, páginas 1203–1204, 2007. http://doi.acm.org/10.1145/1242572.1242766.

[10] Li, H.: *A Short Introduction to Learning to Rank*. IEICE Transactions, 94-D(10):1854–1862, 2011. http://search.ieice.org/bin/summary.php?id=e94-d_10_1854.

[11] Liu, T.-Y.: *Learning to Rank for Information Retrieval*. Enero 2011, ISBN 978-3-642-14266-6.

[12] Lohmann, S., J. Ziegler y L. Tetzlaff: *Comparison of Tag Cloud Layouts: Task-Related Performance and Visual Exploration*. En *Human-Computer Interaction - INTERACT 2009, 12th IFIP TC 13 International Conference, Uppsala, Sweden, August 24-28, 2009, Proceedings, Part I*, páginas 392–404, 2009. https://doi.org/10.1007/978-3-642-03655-2_43.

[13] McNaught, C. y P. Lam: *Using Wordle as a Supplementary Research Tool*. 15, Mayo 2010.

[14] Rennie, J. D. M., L. Shih, J. Teevan y D. R. Karger: *Tackling the Poor Assumptions of Naive Bayes Text Classifiers*. En *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, páginas 616–623, 2003. http://www.aaai.org/Library/ICML/2003/icml03-081.php.

[15] Rose, S., D. Engel, N. Cramer y W. Cowley: *Automatic Keyword Extraction from Individual Documents*, Marzo 2010, ISBN 9780470689646.

[16] Witten, I. H., G. W. Paynter, E. Frank, C. Gutwin y C. G. Nevill-Manning: *KEA: Practical Automatic Keyphrase Extraction*. En *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, páginas 254–255. ACM, New York, NY, USA, 1999, ISBN 1-58113-145-3. http://doi.acm.org/10.1145/313238.313437.

[17] Xu, J. y H. Li: *AdaRank: a boosting algorithm for information retrieval*. En *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, páginas 391–398, 2007. http://doi.acm.org/10.1145/1277741.1277809.

Apéndice A

Evaluaciones

A.1. Tau de Kendall

Tabla A.1: Tabla completa de los índices de correlación de Kendall.

Autores	Lambda MART	AdaRank	RankSVM	Linear Regression	Rake
A	0.2109	0.1723	0.1606	0.1741	0.1723
B	0.4908	0.0223	0.4349	0.4349	0.0223
C	0.0493	0.2084	0.4524	0.2493	0.2084
D	0.3032	0.1524	0.3236	0.3630	0.1524
E	0.3509	0.0368	0.2843	0.3304	0.0368
F	0.3937	0.2702	0.4510	0.4146	0.2702
G	0.2488	-0.0263	0.2096	0.2752	-0.0263
H	0.2921	-0.0034	0.3430	0.3078	-0.0034
I	0.4196	0.2638	0.3492	0.3801	0.2638
J	0.3955	0.0016	0.2874	0.3741	0.0016
K	0.3436	-0.0247	0.3246	0.3889	-0.0247
L	0.4547	0.0700	0.5086	0.5012	0.0700
Promedio:	0.3294	0.0953	0.3441	0.3495	0.0953