Diseño, Implementación, Evolución y Análisis de un Sistema de Recuperación de Información

Loraine Monteagudo García[1] y Tony Raúl Blanco Fernández[1]

Universidad de La Habana, La Habana, Cuba

Resumen En el presente artículo, se describe la implementación de un Sistema de Recuperación de Información basado en el Modelo de Recuperación de Información Vectorial. Se expone el diseño del sistema según cada etapa de la recuperación de información y varias de sus funcionalidades son explicadas. Además, se describen las herramientas empleadas para la programación y la estructura de nuestro proyecto. Se explica la utilización de nuestro sistema y los aspectos más importantes de la interfaz visual del mismo. Luego, se realiza una evaluación del sistema en distintas colecciones de prueba empleando distintas métricas objetivas. Posteriormente, un análisis crítico de las ventajas y desventajas del sistema desarrollado es realizado. Por último, se realizan propuestas de trabajo futuro con el objetivo de mejorar nuestra propuesta.

Keywords: Recuperación de Información, Procesamiento de Lenguaje Natural, Crawling

1. Introducción

El significado del término recuperación de información puede ser muy amplio. Simplemente sacar una tarjeta de crédito de su billetera para que pueda ingresar el número de la tarjeta es una forma de recuperación de información. Sin embargo, como campo de estudio académico, la recuperación de información podría definirse así:

> La recuperación de información (RI) consiste en encontrar material (generalmente documentos) de una naturaleza no estructurada (generalmente texto) que satisface una necesidad de información dentro de grandes colecciones (generalmente almacenadas en computadoras).

Tal como se define de esta manera, la recuperación de información solía ser una actividad en la que solo participaban unas pocas personas: bibliotecarios de referencia, asistentes legales y similares buscadores profesionales. Ahora el mundo ha cambiado y cientos de millones de personas se involucran en la recuperación de información todos los días cuando utilizan un motor de búsqueda web o buscan en su correo electrónico.

La recuperación de información se está convirtiendo rápidamente en la forma dominante de acceso a la información, superando las bases de datos tradicionales. La IR también puede cubrir otros tipos de problemas de datos e información

más allá de los especificados en la definición básica anterior. El término “datos no estructurados”se refiere a datos que no tienen una estructura clara, semánticamente abierta y fácil de usar para una computadora. Es lo opuesto a los datos estructurados, cuyo ejemplo canónico es una base de datos relacional, del tipo que las empresas suelen utilizar para mantener inventarios de productos y registros de personal. RI también se utiliza para facilitar la búsqueda ”semiestructurada”, como encontrar un documento donde el título contiene Java y el cuerpo contiene subprocesos.

El campo de la recuperación de información también cubre el apoyo a los usuarios en la exploración o el filtrado de colecciones de documentos o en el procesamiento posterior de un conjunto de documentos recuperados. Dado un conjunto de documentos, la agrupación es la tarea de crear una buena agrupación de los documentos en función de su contenido. Es similar a organizar los libros en una estantería de acuerdo con su tema. Dado un conjunto de temas, necesidades de información permanente u otras categorías (como la idoneidad de los textos para diferentes grupos de edad), la clasificación es la tarea de decidir a qué clase(s), si corresponde, pertenece cada uno de un conjunto de documentos. A menudo se aborda clasificando manualmente algunos documentos y luego esperando poder clasificar los nuevos documentos automáticamente.

En el presente trabajo se describe el diseño implementación, evolución y análisis de un Sistema de Recuperación de Información. Para ello, se describen todas las etapas del proceso de recuperación de información. Es decir, desde el procesamiento de la consulta hecha por un usuario, a la representación de los documentos y la consulta, el funcionamiento del motor de búsqueda y la obtención de los resultados.

El resto del artículo está organizado de la siguiente manera. La sección 2 describe el diseño completo del sistema según cada etapa de la recuperación de información. La sección 3 presenta las herramientas empleadas para la programación y aspectos más importantes del código. Posteriormente, en la sección 4 se expone la evaluación del sistema empleando distintas métricas y 4 colecciones de prueba distintas. En la sección 5 se realiza un análisis crítico de las ventajas y desventajas del sistema desarrollado. Luego, recomendaciones para trabajos futuros que mejoren la propuesta son expuestos en la sección 6 y en la sección 7 se presentan las conclusiones y un resumen de las características del sistema.

2. Diseño del sistema

El sistema desarrollado cuenta con varias funcionalidades. Primero, dado una consulta, es necesario realizar el preprocesamiento y la representación de la consulta y los documentos. Posteriormente, el Modelo de Recuperación de Información es utilizado para obtener una ordenación de los documentos relevantes. Opcionalmente, el usuario puede participar en la retroalimentación del sistema para mejorar los resultados obtenidos. Además, se implementa la expansión de consultas y el agrupamiento de documentos con el objetivo de mejorar el recobrado del sistema. A su vez, con el objetivo de generar más información para el

sistema se usaron técnicas de *Crawling*. Finalmente, funcionalidades de sistemas de recomendación son usadas con el objetivo de incorporar la recomendación de documentos a nuestro sistema de recuperación de información.

A continuación, cada una de estas funcionalidades son explicada más extensivamente.

2.1. Preprocesamiento de la consulta y los documentos

Para el procesamiento de la consulta del usuario y de los documentos en el corpus es necesario representarlos de una manera lógica para ser procesados por la máquina. Para ello se realizan operaciones de procesamiento textual que permiten estructurarlos. Esto facilita la obtención de información necesaria para identificar los documentos relevantes.

El siguiente preprocesamiento textual es usado para procesar la consulta y los documentos:

Eliminación de los signos de puntuación: Los signos de puntuación (`,` `.` `;` `:`) son eliminados, así como otros caracteres especiales (`#<>*+`) que no se piensa que aporten información en la tarea de recuperación de información.

Minúsculas: Las mayúsculas/minúsculas no aportan información relevante, así que todo el texto es representado en minúsculas.

Tokenización: El texto es convertido en una secuencia de *tokens*, que es una cadena de caracteres que tiene un significado coherente en el idioma usado.

Eliminación de stopwords: los stopwords son las palabras que no proveen información útil para la clasificación de un texto.

Stemming: Es un método para reducir una palabra a su raíz o (en inglés) a un stem. Hay algunos algoritmos de stemming que ayudan en sistemas de recuperación de información, en nuestro sistema se el algoritmo de Porter.

2.2. Representación de la consulta y los documentos

El siguiente paso, después del preprocesamiento textual, es la construcción de índices. Un término indexado es una palabra cuya semántica ayuda a definir el tema principal de un documento, brindando un resumen de su contenido. Generalmente se usan como términos indexados las palabras contenidas en el documento.

La recuperación de información basada en términos indexados tiene como base fundamental la idea de que la semántica de documentos y las necesidades informativas de los usuarios pueden ser expresadas a través de un conjunto de términos indexados.

Existen diferentes métodos de representación de documentos. Uno de los más empleados son las matrices de asociación. En ellas se representan en una dimensión, ya sea filas o columnas, los documentos y en la otra los términos o palabras presentes en el vocabulario resultado del preprocesamiento textual de los documentos en el corpus. Cada celda de esta matriz indica la presencia de un término en un documento. Esta estructura de datos permite realizar operaciones

de una manera relativamente eficiente, sin embargo, impone grandes costos de almacenamiento, pues por cada documento hay un vector con todos los términos del vocabulario y es probable que la mayor parte de ellos no esté contenida en todos los documentos. Esto origina que la matriz sea esparcida, es decir, la mayoría de los valores son cero.

Otra de las estructuras de datos más empleada en la actualidad son los índices invertidos. Cada término apunta a una lista de los documentos en donde está incluido. De esta manera se hace un uso más eficiente del espacio de almacenamiento, sin embargo, se dificulta el uso eficiente de dichos índices.

Como los modelos clásicos de recuperación de información consideran que cada documento está descrito por un conjunto de palabras clave (términos indexados) se hace necesario la búsqueda eficiente de los términos dentro de un documento, razón por la cual los índices invertidos no fue la estructura utilizada para la implementación en nuestro sistema. En su lugar, se realizó una versión de las matrices de asociación usando una técnica muy usada cuando se trabajan con matrices esparcidas: la matriz se representa como una lista de diccionarios donde cada elemento de la lista serían las filas, las llaves de los diccionarios las columnas de la matriz y su valor el valor que ocuparía esa posición. Solo se almacenan los valores distintos de cero. Por lo tanto, de no existir un valor en el diccionario este se asume que es cero. Luego, nosotros tenemos listas de documentos en los que cada uno de ellos contiene un diccionario con los términos que están presentes en el documento como llave y la frecuencia de ellos como valor.

La estructura utilizada contiene además más información. Se realiza un mapeo entre términos y enteros, haciéndole corresponder a cada término del conjunto de documentos un valor entero y único que lo identifica. Se guarda además más información útil para las siguientes etapas de recuperación de información, como la cantidad de documentos en los que está presente un término.

2.3. Modelo de Recuperación de Información

Un modelo de recuperación de información (MRI) es un cuádruplo $[D, Q, F, R(q_j, d_j)]$ donde:

D es un conjunto de representaciones lógicas de los documentos de la colección.
Q es un conjunto compuesto por representaciones lógicas de las necesidades del usuario. Estas representaciones son denominadas consultas.
F es un *framework* para modelar las representaciones de los documentos, consultas y sus relaciones.
R es una función de ranking que asocia un número real con una consulta $q_j \in Q$ y una representación de documento $d_j \in D$. La evaluación de esta función establece un cierto orden entre los documentos de acuerdo a la consulta.

En la literatura se identifican tres modelos clásicos de recuperación de información: el modelo booleano, el vectorial y el probabilístico.

El modelo de recuperación de información usado en este sistema es el MRI Vectorial ya que este es simple, rápido, y en algunos casos, brinda mejores resultados en la recuperación de información que el resto de los MRI clásicos. Sin

embargo, sabemos que esto no significa que sea el mejor modelo; no existe un modelo general que dé solución a todos los problemas.

De manera general, todos los modelos de recuperación de información tienen alguna noción de peso. Esta definición siempre está relacionada con la importancia de un término en un documento. Cada documento va a tener asociado un vector de términos indexados con la información del peso de cada uno obtenida a partir de una función. Es común a todos los modelos que si un término no aparece en un documento su peso sea cero.

En el modelo vectorial, el peso de un término t_i en el documento d_j está dado por:

$$w_{i,j} = tf_{i,j} \times idf_i$$

Sea $freq_{i,j}$ la frecuencia del término t_i en el documento d_j. Entonces la frecuencia normalizada $f_{i,j}$ del término t_i en el documento d_j ($tf_{i,j}$) se calcula como:

$$tf_{i,j} = \frac{freq_{i,j}}{max_l freq_{l,j}}$$

donde el máximo (término l) se calcula sobre los términos del documento d_j.

Por otro lado, sea N la cantidad total de documentos en el sistema y n_i la cantidad de documentos en los que aparece el término t_i. La frecuencia de ocurrencia de un término t_i dentro de todos los documentos de la colección idf_i (del inglés *inverse document frequency*) está dada por:

$$idf_i = log\frac{N}{n_i}$$

La medida $tf_{i,j}$ es una medida de similitud intra-documento. Se considera la frecuencia de cada término en un documento dividido entre la frecuencia máxima de ese mismo documento para normalizar esta medida. Esto se hace con el objetivo de evitar valores de frecuencia sesgados por la longitud del documento.

Por otro lado, idf_i es una medida inter-documentos. Además de la medida de frecuencia de términos es importante analizar la frecuencia de ese término en todos los documentos de la colección. Un término que aparezca en pocos documentos de la colección tiene mayor valor frente a una consulta pues permite discriminar una mayor cantidad de documentos.

En el caso del cálculo de pesos para la consulta se introduce la frecuencia de los términos con una constante de suavizado (a). Esta permite amortiguar la variación en los pesos de términos que ocurren poco, para evitar, por ejemplo, grandes saltos entre la frecuencia de un término que aparece una vez a otro que aparece dos veces. Los valores más usados son 0.4 y 0.5, en nuestro sistema usamos el término de 0.4. Sea además $freq_{i,q}$ la frecuencia del término t_i en el texto de consulta q, el peso de una consulta está dado por la siguiente fórmula:

$$w_{i,q} = (a + (1-a)\frac{freq_{i,q}}{max_l freq_{l,q}}) \times log\frac{N}{n_i}$$

Una vez definidos los pesos de los documentos y las consultas solo falta la función de ranking para completar la definición del modelo vectorial. Esta función permite tener una ordenación por relevancia de los documentos y se basa en la similitud entre la consulta y los documentos empleando el coseno del ángulo comprendido entre los vectores documentos d_j y la consulta q:

$$sim(d_j, q) = \frac{\overrightarrow{d_j} \cdot \overrightarrow{q}}{|\overrightarrow{d_j}| \cdot |\overrightarrow{q}|} = \frac{\sum_{i=1}^{n} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \times \sum_{i=1}^{n} w_{i,q}^2}$$

Luego de obtener este ranking, se ordenan los documentos de acuerdo a su medida de similitud de mayor a menor, ya que documentos más similares tienen una mayor similitud. Para la recuperación de los documentos, puede establecerse un umbral de similitud y recuperar los documentos cuyo grado de similitud sea mayor que este umbral. En nuestro sistema se usó el umbral 0.3, que fue decidido de manera experimental para garantizar la recuperación de la mayor cantidad de documentos.

2.4. Retroalimentación

A veces para los usuarios es díficil plantear las consultas de modo que expresen sus necesidades informativas. Además, muchas veces los SRI no logran dar respuesta a la necesidad de información del usuario. Es por esto que una idea interesante podría involucrar al usuario en un proceso de retroalimentación que permita obtener mejores resultados.

El algoritmo clásico de retroalimentación, qué es además el usado por nuestro sistema consiste en la realización de los siguientes pasos:

1. El usuario plantea la consulta.
2. El sistema devuelve un conjunto de documentos.
3. El usuario selecciona de estos documentos los que considera relevantes o no.
4. El sistema obtiene una mejor representación de las necesidades del usuario usando esta información.
5. Se regresa al paso 2.

Existen algoritmos específicos de retroalimentación para cada uno de los MRI clásicos en dependencia de las características de cada uno de ellos. La retroalimentación en el MRI Vectorial considera que los vectores de los documentos relevantes (a una consulta) son similares y que tienen una diferencia notable con los vectores de los documentos relevantes. Por lo tanto, la idea principal del algoritmo de retroalimentación consiste en encontrar un vector consulta $\overrightarrow{q}$ que maximice la similitud de los documentos relevantes mientras minimice a similitud con los documentos no relevantes.

Asumiendo que se conocen C_r el conjunto de documentos relevantes y C_{nr} el conjunto de documentos no relevantes la función del algoritmo de retroalimentación es encontrar el vector $\overrightarrow{q}_{opt}$ que representa el vector promedio del conjunto de documentos relevantes y no relevantes respecto a la consulta expresado como:

$$\overrightarrow{q}_{opt} = max[sim(\overrightarrow{q}, C_r) - sim(\overrightarrow{q}, C_{nr})]$$

Utilizando el cálculo del coseno del ángulo entre los vectores de la consulta y los documentos como medida de similitud, resulta:

$$\overrightarrow{q}_{opt} = \frac{1}{|C_r|} \sum_{\overrightarrow{d_j} \in C_r} \overrightarrow{d_j} - \frac{1}{|C_{nr}|} \sum_{\overrightarrow{d_j} \in C_{nr}} \overrightarrow{d_j}$$

En un SRI real, se tiene una consulta y solo se conoce parcialmente el conjunto de los documentos relevantes y no relevantes. El algoritmo de Rocchio ofrece una alternativa para el cálculo de esta consulta ideal a partir del conocimiento de algunos documentos relevantes y no relevantes. Además, considera la influencia de la consulta original y otorga un peso a cada componente de la fórmula. De esta manera podemos ajustar a conveniencia la importancia de cada uno. Dado D_r y D_{nr} el conjunto de documentos relevantes y no relevantes conocidos respectivamente y α, β, γ los pesos establecidos para cada término en la consulta el algoritmo de Rocchio propone la siguiente fórmula:

$$\overrightarrow{q}_m = \alpha \overrightarrow{q} + \frac{\beta}{|D_r|} \sum_{\overrightarrow{d_j} \in D_r} \overrightarrow{d_j} - \frac{\gamma}{|D_{nr}|} \sum_{\overrightarrow{d_j} \in D_{nr}} \overrightarrow{d_j}$$

Los valores de cada constante dependen del contexto de aplicación. Si tenemos muchos documentos, β y γ pueden tener valores grandes. Se puede asumir que es más importante la información obtenida de los documentos relevantes que de los no relevantes, por lo que β (retroalimentación positiva) puede ser mayor que γ (retroalimentación negativa). Los valores usados en nuestro sistema son $\alpha = 1$, $\beta = 0{,}75$ y $\gamma = 0{,}15$ por ser valores comúnmente usados.

En el sistema implementado los vectores guardados en el algoritmo de Rocchio, tanto de la consulta como de los documentos ($\overrightarrow{q}$, $\overrightarrow{d_j}$), son representados por la frecuencia de sus términos.

Una de las desventajas de los algoritmos de retroalimentación es que, en la práctica, pocos usuarios participan en ella. Por lo tanto, existen otros métodos de retroalimentación que omiten la intervención del usuario. En nuestro sistema implementamos además la pseudo-retroalimentación para estos casos. Con este método se seleccionan como documentos relevantes los k primeros documentos del ranking dado como resultado del sistema y luego se aplica la retroalimentación. En nuestra propuesta el valor tentativo de k es 10, sin embargo, probablemente una búsqueda más refinada de este parámetro sea necesaria.

Otra de las limitaciones de la retroalimentación es que las consultas que se generan son muy grandes e inificientes para los algoritmos de recuperación. Por esto, una reducción de las dimensiones de este vector es realizada, solo cogiendo los valores que sean mayores de $\epsilon = 0{,}05$. Este es otro parámetro del sistema que necesita ser mejor determinado.

2.5. Expansión de consultas

Otra variante para lograr la retroalimentación del Sistema de Información es adecuar mejor la consulta al conjunto de documentos relevantes esperados. La expansión de consultas es una variante muy atractiva que permite mejorar las respuestas de los modelos y que además, se puede usar conjuntamente con la retroalimentación en otros modelos. Una de sus principales ventajas es que no requiere la intervención del usuario. En nuestro sistema, es usada conjuntamente con la retroalimentación ofreciendo alternativas de la consulta original al usuario.

Los métodos de expansión de consultas pueden ser clasificados atendiendo a si la información que emplean es local o externa al modelo

- Análisis global
  - Utilizan fuentes externas como tesauros u otras bases de conocimientos.
  - Es la forma más común de expansión
- Análisis local
  - Analizan los documentos en la colección

Con el objetivo de mejorar el funcionamiento de nuestro sistema usamos el análisis global como fuente de información de la expansión de consultas. Para esto, usamos Wordnet como tesauro, que dado una palabra nos da información concerniente a dicha palabra: sinónimos, antónimos y otras relaciones semánticas más avanzadas como hipernónimos, hiponónimos, etc. Para la implementación de nuestro sistema consideramos la sustitución por sinónimos y hipernónimos. En semántica lingüística, se denomina hiperónimo a aquel término general que puede ser utilizado para referirse a la realidad nombrada por un término más específico. Por ejemplo, ser vivo es hiperónimo para los términos planta y animal.

Nuestro sistema combina el uso de tesauros con la ponderación de términos. Se considera *idf* que es una medida inter-documentos para determinar aquellos términos que están poco presentes en el corpus, y de esos términos se intenta hacer una sustitución (ya sea por sinónimos o por hipernónimos). Para esto se determina un umbral, que tentativamente es $log\frac{N}{20}$, donde N es la cantidad total de documentos del corpus. Los términos que tengan un *idf* mayor q el umbral determinado quiere decir que aparece en menos de 20 documentos, y son los elegidos para ser posiblemente sustituidos. Luego de que estos términos sean determinados se procede a la sustitución de estos por sinónimos o hipernónimos cuyo *idf* sea mayor que el umbral anteriormente determinado.

Además, se realizó la implementación de un corrector ortográfico. Para esto se cuenta con una lista de palabras en inglés (que cuenta con 236736 palabras). Para cada una de los términos de la consulta se calcula la distancia de edición de Levenshtein con respecto a las palabras en inglés. La distancia de edición es el número de caracteres que necesitan ser sustituidos, insertados o eliminados para transformar una palabra `s1` en `s2`. La palabra en inglés que tenga la mínima distancia de edición con cada término de la consulta es la palabra escrita correctamente. Este término correctamente escrito es sustituido en la consulta original.

2.6. *Crawling*

Un *crawler* es un agente del tipo bot que recorre recursivamente la *World Wide Web* bajo algún orden predeterminado, y que recopila información acerca de los documentos que encuentra y su estructura de vínculos.

Según sea el caso, un *crawler* puede ser programado para trabajar con un propósito general o sobre un dominio específico. En nuestro caso, el *crawler* implementado recoge información definida sobre el dominio de Wikipedia. Dado un `n` especificado recoge información del resumen de `n` artículos de Wikipedia siguiendo el siguiente algoritmo:

1. Mientras la cantidad de artículos extraídos sea menor que n:
2. Si la lista de urls es cero, entonces se agrega un url random de Wikipedia (esto es posible mediante el link https://en.m.wikipedia.org/wiki/Special:Random)
3. Se extrae un url de la lista de urls.
4. El título del documento y el resumen del artículo de wikipedia es guardado y son extraídos todos los enlaces del documento que estén bajo el dominio de `https://en.m.wikipedia.org/wiki`.
5. Añade a la lista de urls cada uno de los enlaces extraídos, siempre que no haya sido extraído antes.

La información extraída de los artículos de Wikipedia son almacenados en un diccionario donde la llave es el título del artículo y el valor es el contenido del resumen del mismo. Esta infomación es almacenada en un `json` para ser fácilmente parseada e indexada, con el objetivo de ser utilizada como corpus adicional del sistema.

2.7. Agrupamiento de documentos

Los algoritmos de agrupamiento de documentos obtienen una división de un conjunto de documentos, de modo que los documentos similares estén en el mismo grupo (clúster), mientras que los documentos disimilares estén en grupos distintos.

La cantidad de clases o grupos es un parámetro de los métodos de agrupamiento que debe ser estimado. Un método muy común es el método del codo. Para determinar la cantidad óptima de grupos se usa como métrica la suma de las diferencias al cuadrado entre cada elemento y el centroide del grupo asignado a través del método de agrupamiento. Intuitivamente, la cantidad ideal es justo el valor del eje x (cantidad de grupos) en el punto con el mayor cambio en la pendiente de la función. Muy pocos grupos generan grandes diferencias y muchos grupos no generalizan lo suficiente. La respuesta deseada equilibra ambos aspectos. Este método fue el usado para estimar la cantidad de grupos en cada uno de los corpus usados.

Existen varias maneras de representar los documentos, a través de vectores binarios, vectores reales ($tf \times idf$) o *Word Embeddings*. En nuestro algoritmo los documentos fueron representados a partir de vectores reales, calculando $tf \times idf$,

debido a la simplicidad de esta representación, ampliamente usada en nuestro sistema. A su vez, existen varias medidas de similitud, la usada para los vectores reales es el producto escalar y la distancia euclidiana entre los vectores. La última fue la elegida para nuestro sistema.

Según el tipo de estructura impuesta sobre los datos los algoritmos de *clustering* se clasifican como jerárquicos y particionados. Los algoritmos de agrupamiento particionados obtienen una partición simple de N documentos en un conjunto de K grupos. El algoritmo usado fue *K-means*, pertenece a este último grupo. Fue elegido debido a su simplicidad, su eficiencia y su amplio uso.

La definición general de los algoritmos de agrupamiento es: Dado un conjunto de documentos D = {d_1, d_2, ..., d_N}, la cantidad de grupos K deseados y una función objetivo que evalúa la calidad del agrupamiento encontrar una asignación $P : D- > w_1, w_2, ..., w_k$ que minimice la función objetivo, tal que, ningún w_i esté vacío.

En el caso de *K-means* se selecciona un representante de cada grupo, que es el vector promedio de todos los pertenecientes al grupo. Luego, tiene como función objetivo el cuadrado de la distancia Euclidiana media de los vectores del grupo.

$$RSS_k = \sum_{\overrightarrow{x} \in \omega_k} |\overrightarrow{x} - \overrightarrow{u}(\omega_k)|^2$$

$$RSS = \sum_{k=1}^{K} RSS_k$$

El algoritmo de *K-means* consiste en:

- Seleccionar K documentos aleatoriamente {s_1, s_2,..s_k} como centroides.
- Mientras no se cumpla una condición de parada:
  - Asignar a cada documento d_i al cluster w_j tal que la distancia entre d_i y s_j sea la menor respecto a los demás clusters.
  - Actualizar los centroides de cada cluster.

Es necesario notar, que para la realización de nuestro sistema no implementamos desde cero el algoritmo de *K-means*, si no que usamos una implementación existente, en concreto, la desarrollada en `sklearn`.

Clustering tiene diferentes aplicaciones en recuperación de información, la usada por nosotros explota la hipótesis de la agrupación para mejorar los resultados de búsqueda, basándose en una agrupación de toda la colección. Identificamos un conjunto inicial de documentos que coinciden con la consulta, pero luego agregamos otros documentos de los mismos clusters de los documentos retornados, incluso si tienen poca similitud con la consulta. Por ejemplo, si la consulta es carro y se toman varios documentos de carro de un cluster de documentos de automóvil, entonces podemos agregar documentos de este grupo que usen términos distintos a carro (automóvil, vehículo, etc.). Esto puede aumentar el recobrado, ya que un grupo de documentos con una gran similitud mutua suele ser relevante en su conjunto.

En concreto, retornamos para el usuario 10 documentos que estén en el mismo cluster que el primero document retornado. Estos documentos son puestos después de los 10 primeros documentos devueltos por nuestro MRI. Notar que para esto es necesario guardar todos los documentos que están en todos clusters.

2.8. Recomendación de documentos

Existe una amplia clase de aplicaciones web que implican predecir las respuestas de los usuarios a las opciones. Esta facilidad se denomina sistema de recomendación.

La aplicación usada de los sistemas de recomendación en nuestro sistema de recuperación de información fue la recomendación de documentos. De esta manera, devolvemos documentos que pueden resultar interesantes a nuestros usuarios basados en las consultas realizadas por este. Se almacenan como documentos en los que el usuario muestra interés el primero que es retornado en la búsqueda de una consulta y aquellos que el usuario marca como relevantes en la retroalimentación.

Para el desarrollo de un sistema de recomendación es necesario la creación de un perfil de ítem para cada ítem. El perfil de ítem es un conjunto de características que matemáticamente es interpretado como un vector de características. Cada componente del vector representa una característica y puede tener solo valores binarios o reales. En nuestro caso, no es inmediatamente aparente las características que pueden ser utilizadas para representar a nuestros ítems, que son los documentos. La medida $tf - idf$ de cada palabra en el documento tantas veces usada y computada por nuestro sistema es el usado para representarlos.

Además, se definen los perfiles de usuario que es un conjunto de características que representan los intereses del usuario. En nuestro sistema de recomendación, que es sencillo y es ejecutado localmente, no contamos con un concepto para un usuario. Se considera que el usuario es único, al no contar con un sistema para la autentificación que permita que existan varios.

Los sistemas de recomendación utilizan varias tecnologías diferentes. Podemos clasificar estos sistemas en dos grandes grupos.

- Los sistemas basados en contenido se centran en las propiedades de los elementos. La similitud de los elementos se determina midiendo la similitud en sus propiedades. Por ejemplo, si un usuario de Netflix ha visto muchas películas de vaqueros, recomiende una película clasificada en la base de datos como del género "vaquero".
- Los sistemas de filtrado colaborativo se centran en la relación entre usuarios y elementos. La similitud de elementos está determinada por la similitud de las calificaciones de esos elementos por parte de los usuarios que han calificado ambos elementos.

Sin embargo en la práctica, funcionan mejor los sistemas de recomendación híbridos, en los que se emplean varias técnicas de filtrado y se combinan predicciones y elementos de varios enfoques. Por esto elegimos un enfoque híbrido para la realización de la recomendación de documentos.

Una práctica común es proponer alternativas que consideran efectos locales y globales. El algoritmo seguido por nuestro sistema es el siguiente:

- Definir la similitud S_{ij} de los ítems i y j.
- Seleccionar los k vecinos más cercanos $N(i;x)$, es decir, los ítems más similares a i que fueron evaluados por x.
- Estimar el rating $\hat{r}_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} S_{ij}(r_{xj} - b_{xj})]}{\sum_{j \in N(i;x)} S_{ij}}$

Donde:

μ = media de los *ratings* de todos los ítems.
b_x = desviación del *rating* del usuario x = (media del *rating* del usuario x) - μ.
b_i = (media del *rating* del ítem i) - μ.
Predictor Baseline: $b_{xi} = \mu + b_x + b_i$

Es necesario notar que existen componentes en la fórmula que suponen la existencia de varios usuarios que no es consistente con nuestro sistema, que asume que el usuario de la aplicación es único. La medida b_x que representa la desviación del *rating* de usuario no tiene sentido porque los *ratings* de todos los ítems son realizados por el mismo usuario, por lo que esta desviación siempre es cero.

Para seleccionar los k vecinos más cercanos $N(i;x)$ se usa para ganar en simplicidad la idea de usar los clusters calculados con el algoritmo *K-meas*. Por lo tanto, se seleccionan todos los documentos que estén en el mismo cluster que el documento i que fueron evaluados por x.

La única componente que falta por definir en la fórmula es entonces la medida de similitud S_{ij} de los ítems i y j. Dados los *ratings* de dos documents A y B denominados r_A y r_B respectivamente, dos medidas de similitud fueron consideradas:

Similitud de Jaccard: El problema principal de esta medida es que ignora los valores de los ratings, es una medida binaria:

$$sim(A, B) = \frac{|r_A \cup r_B|}{|r_A \cap r_B|}$$

Similitud de coseno: En este caso, el problema son los vectores faltantes, que el coseno toma como cero, por lo que probablemente usuarios con una misma cantidad de documentos evaluados sean similares, independientemente de sus valores de ratings.

$$sim(A, B) = cos(r_A, r_B) = \frac{r_A \cdot r_B}{|r_A||r_b|} =$$

Ambas medidas fueron consideradas, pero los mejores resultados se obtuvieron con la medida de Jaccard. El problema de tratar los valores que faltan como cero de la medida del coseno resultó ser más grave que el problema de ignorar los valores de los ratings de la medida de Jaccard, que por otro lado, no resulta nada grave ya que en primera instancia los valores de los ratings son tomados como valores binarios.

3. Implementación

El sistema fue implentado en `Python 3.7.3`. Las principales bibliotecas utilizadas y sus usos fueron:

- `nltk`: Para el procesamiento de texto. Se usa para tokenizar, para realizar stemming, para la eliminación de los *stopwords* mediante la lista de *stopwords* presente en `nltk`. Además se usa como corrector ortográfico en el uso de la distancia de edición de Levenshtein y su lista de palabras de inglés. A su vez, se usa la implementación de WordNet de `nltk`, que es usado como tesauro para la expansión de consultas. Para el total funcionamiento del sistema además de la instalación de nltk es necesario descargar datos adicionales, `nltk_data`.
- `gensim`: Se usa el `Diccionary` localizado en `gensim.corpora` que encapsula el mapeo entre las palabras normalizadas y sus ids en entero. Es la estructura utilizada para realizar el *indexing* de los corpus. Se inicializa con una lista de documentos cuyo texto representado por *tokens* (listas de palabras). Tiene un conjunto de atributos y métodos, como `doc2bow`, que dado el índice de un documento devuelve una lista de tuplas, una componente representa el índice de un token y la otra la frecuencia de dicho token en el documento. Solo se devuelven los tokens que tienen al menos una ocurrencia en el texto. Por otro lado, también es usado en el MRI implementado el atributo `dfs` que retorna cuantos documentos contienen un *token* en específico.
- `sklearn`: Es un módulo para aprendizaje de máquinas para Python. Su implementación del algoritmo de *K-means* es utilizado.
- `yellowbrick`: Es un módulo para el análisis visual y herramientas de diagnóstico para facilitar la selección de características, la selección de modelos y el ajuste de parámetros para el aprendizaje de máquinas. Fue usado el método `KElbowVisualizer` como implementación del método del codo para la estimación del valor óptimo de K en el algoritmo de *K-means*.
- `streamlit`: Fue usado para la realización de la interfaz visual del proyecto.

El link de Github del proyecto, donde puede ser clonado es:
`https://github.com/lorainemg/information-retrieval-system.git`

3.1. Estructura del proyecto

La estructura del proyecto es la siguiente:

- `doc/`: carpeta donde se encuentra la documentación de este proyecto (este archivo).
- `resources/`: carpeta donde se encuentra los recursos usados y generados por el sistema.
  - `cluster/`: Contiene datos (la mayoría de ellos binarios generados con `pickle`) que son producidos por el algoritmo de *clustering*. Estos archivos tienen el objetivo de mejorar el rendimiento del sistema guardando una instancia del algoritmo de *K-means* ya entrenado y una distribución de

los clusters formados para cada uno de los corpus usados. Inicialmente esta carpeta está vacía, estos datos se llenan después de ejecutar por primera vez el algoritmo de *clustering* en cada uno de los corpus.

- `corpus/`: Se guardan los corpus utilizados por el sistema. Todos tienen una estructura similar, un archivo o más que corresponde a los documentos, cada uno de ellos tienen información concerniente a las palabras del documento, título, y algunos cuentan además con el autor de ellos y su origen. Otro archivo contiene consultas pregeneradas y otro la relación existente entre cada una de las consultas y los documentos, es decir, los documentos relevantes para cada consulta. Hay un total de 4 corpus de este estilo: `cisi`, `cran`, `lisa` y `npl`. Adicionalmente, está el corpus generado por el `crawler`, guardado en `json` para que sea fácilmente parseable, `wiki_docs.json`, que solo contiene los títulos y el texto de varios artículos de Wikipedia.
- `indexed_corpus`: De una forma similar a lo que se hace en la carpeta `cluster`, es guardada información sobre los distintos corpus indexados que es generado después de la primera ejecución del sistema.
- `queries`: Los nuevos vectores de las consultas creadas por el algoritmo de retroalimentación de Rocchio son guardadas persistentemente.
- `ratings`: Los *ratings* usados por el algoritmo de recomendación de documentos es guardado persistentemente.
- `test`: Varios `json` son guardados concernientes al parseo de los archivos de los corpus que se refieren a las consultas y los documentos relevantes para cada consulta, que son utilizados en el proceso de evaluación del sistema. Estos archivos son generados después de la primera ejecución de la evaluación.

- `src/`: Se encuentra todo el código escrito en Python del sistema de recuperación de información desarrollado. A continuación se expone el funcionamiento de cada uno de los módulos usados:
  - `corpus/`: Contiene las clases concernientes al parseo y procesamiento de los distintos corpus. Se define una clase base `CorpusAnalyzer` en `corpus.py` que es abstracta, cuenta con la implementación de todos los métodos útiles para el funcionamiento del sistema relacionado con el procesamiento y representación de los documentos. Aquí se implementa, por ejemplo, el preprocesamiento textual de los corpus y la creación de los indices. El método abstracto se refiere al parseo del los documentos del corpus, su implementación depende del corpus a utilizar por lo cual se definen 5 más clases concernientes a cada uno de los corpus que sobreescriben la definición de este método.
  - `models/`: En `model.py` se define una clase base para la implementación de un modelo de recuperación de información donde el método `ranking_function` no está implementado. Luego, en `vector_mri.py` se encuentra la implementación de este método para el MRI vectorial.
  - `tests/`: Se encuentra la definición de varias clases para el parseo de los archivos de los distintos corpus para la realización de los tests (las consultas y el orden de la relevancia de los documentos para cada una

de estas consultas). Además, en `test_manager.py`, que es la clase usada para realizar los tests en los distintos corpus, se encuentra implementado el método general `test_model.py` que calcula con una o varias medidas de evaluación definidas el rendimiento del sistema.
  - `visual/`: Se implementan algunas utilidades usadas en la interfaz visual del sistema.
  - `clustering.py`: Se define la clase encargada de la realización del algoritmo de *clustering* del sistema, en nuestro caso, se hace uso de *K-means*.
  - `clawler.py`: Se define la clase que recorre n artículos de Wikipedia y recopila información acerca de ellos.
  - `document_recommendation.py`: Se define la clase encargada de la realización de la recomendación de documentos.
  - `evaluation.py`: Se implementan las diferentes medidas de evaluación del sistema.
  - `feedback.py`: Se implementa el algoritmo de retroalimentación de Rocchio.
  - `query_expansion.py`: Se definen los distintos métodos necesarios para la expansión de consultas.
  - `query.py`: Se define una clase encargada de parsear una consulta, realizar el procesamiento y su representación como un vector.
  - `streamlit.py`: Archivo donde se definen los métodos principales para la ejecución del proyecto mediante `streamlit`.
  - `main.py`: Módulo para hacer queries interactivas mediante la consola
  - `system.py`: Módulo principal donde se implementa el sistema de recuperación de información, donde están encapsuladas casi todas las funcionalidades del proyecto.
  - `test.py`: Módulo donde se evalúan los diferentes corpus con distintas medidas de evaluación.
  - `tools.py`: Archivo donde se implementan las algunas definiciones útiles del sistema, en este caso, la definición de lo que es un Documento.
  - `utils.py`: Se implementan varias funciones que son útiles para varios archivos, como preprocesamientos textuales y el calculo de $tf - idf$.
- `requirements.txt`: Documento con todos los requerimientos del sistema. Pueden ser instalados con `python -m pip install -r requirements.txt`.

3.2. Ejecución del sistema

Existen dos formas de ejecución del sistema: la realización dinámica de consultas, ya sea mediante `streamlit_start.py` o `main.py` y la ejecución del escenario de evaluación que está definido en 4 en `test.py`.

Para la realización dinámica de las consultas se realizó una interfaz visual en `streamlit`, para su ejecución es necesario la ejecución del siguiente comando: `cd src/; streamlit run streamlit_start.py`. Después, un browser será abierto. Arriba tendrá un *textbox* donde se escribirá la consulta, los documentos relevantes para dicha consulta son buscados después de presionar *Enter*. A la izquierda existen dos opciones, una para configurar la cantidad de resultados

de la consulta que se quiere ver y otra para el corpus que se usará para buscar documentos relevantes. Además, existe una opción para ver documentos recomendados, mostrando un conjunto de documentos que pueden ser interesantes para el usuario (basados en el resultado del sistema de recomendación). Una vez ejecutada la consulta un conjunto de documentos son mostrados, (su título y las primeras líneas del texto de los mismos). A la derecha se encuentra un *Checkbox* para seleccionar los documentos que se consideran relevantes, una vez seleccionados, se presiona el botón `Send Feedback` localicado al final de la lista de documentos para realizar la retroalimentación. Además, un conjunto de términos relacionados con la consulta original es presentada (aquí es donde se muestra el resultado de la expansión de consultas).

Además, en `main.py` se pueden realizar de forma interactiva consultas a través de la consola de forma similar a como se hace en la interfaz visual.

Para la realización dinámica de consultas de nuestra aplicación un nuevo corpus fue usado, que es la unión de todos los corpus utilizados. De esta manera, el usuario puede realizar consultas en todos los corpus disponibles. Además, se cuenta con el corpus de Wikipedia, obtenido a través del *Crawler*, aunque la versión de este corpus que fue guardada contiene muy pocos documentos como para ser considerado interesante.

Por otro lado, la ejecución de las pruebas se pueden realizar mediante el comando `cd src; python test.py`.

4. Evaluación del sistema

Con el objetivo de analizar el rendimiento de nuestro sistema este fue probado utilizando distintos corpus, que contenían varias consultas de prueba con los documentos que son relevantes. Algunas contenían además un ranking de relevancia entre estos documentos y otros no. De todas formas, este ranking no fue tomado en cuenta para la evaluación del sistema de recuperación de información.

Las colecciones de prueba usadas fueron:

Cranfield: Contiene 1400 documentos y 365 consultas.
CISI: Contiene 1460 documentos y 112 consultas.
LISA: Sus iniciales significan Resúmenes de bibliotecas y ciencias de la información (del inglés **L**ibrary and **I**nformation **S**cience **A**bstracts). Contiene 5872 documentos y 35 consultas.
NPL: Contiene 11429 documentos y 93 consultas.

4.1. Medidas de evaluación

Para describir las medidas de evaluación implementadas comencemos con las siguientes definiciones:

RR: conjunto de documentos recuperados relevantes.
RI: conjunto de documentos recuperados irrelevantes.
NR: conjunto de documentos no recuperados relevantes.

NI: conjunto de documentos no recuperados irrelevantes.

La **Precisión** es una de las medidas fundamentales. La precisión tiende a decrecer cuando la cantidad de documentos recuperados aumenta. Calcula lo fracción de los documentos recuperados que son relevantes:

$$P = \frac{|RR|}{|RR \cup RI|}$$

Por otro lado, tenemos el **Recobrado**, que es fundamental en los procesos de recuperación de información. En contraposición a la precisión el recobrado aumenta a medida que incorporamos más documentos a la respuesta, pues es cada vez más probable que los elementos del conjunto de documentos relevantes estén contenidos en nuestra respuesta. Esto hace que siempre sea posible tener el mayor valor de recobrado, 1. Esto se lograría devolviendo la colección completa aunque, lógicamente, no es una solución factible. Representa la fracción de los documentos relevantes que fueron recuperados.

$$R = \frac{RR}{|RR \cup NR|}$$

Con el objetivo de lograr una compensación entre la precisión y el recobrado se define la medida **F**. Permite enfatizar la precisión sobre el recobrado y viceversa.

$$F = \frac{(1+\beta^2)PR}{\beta^2 P + R} = \frac{1+\beta^2}{\frac{1}{P} + \frac{\beta^2}{R}}$$

La medida **F1** es un caso particular de la medida F en la que la precisión y el recobrado tienen igual importancia.

$$F_1 = \frac{2PR}{P+R} = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

Otras medidas de evaluación relacionadas con las anteriormente expuestas es la **R-Precisión**, que se define como la precisión en la pocisión R del *ranking* de documentos relevantes a una consulta dada para la cual existen ***R documentos relevantes***. De manera análoga se puede aplicar este criterio al recobrado y la medida F.

Una de las principales desventajas de estas medidas es que no tienen en cuenta los documentos irrelevantes, y con este objetivo se introduce la **Proporción de fallo (*Fallout*)**. Esta medida tiene en cuenta la cantidad de documentos irrelevantes y el ranking.

$$Fallout = \frac{|RI|}{|RI \cup NI|}$$

4.2. Resultados obtenidos

A pesar de que todas estas medidas fueron implementadas en nuestro sistema, para medir su evaluación fueron consideradas 3: la precisión, el recobrado y la medida f1.

Como en nuestras colecciones de prueba se tenían la cantidad de documentos relevantes para cada consulta y el resultado del Modelo de Recuperación de Información es un ranking entre los documentos relevantes (existe una ordenación entre ellos) se utilizaron medidas que trabajan con el ranking de los documentos. Es decir, se calculó para cada consulta la *R-Precisión*, *R-Recobrado* y *R-F1*, donde R es la cantidad conocida de documentos relevantes de la consulta. El promedio de cada una de estas mediciones de cada consulta individual es el calculado para evaluar el rendimiento del sistema.

En el Cuadro 1 se muestran los resultados obtenidos por el sistema.

Corpus	**Precisión**	**Recobrado**	**F1**
Cranfield	0.139	0.139	0.139
CISI	0.085	0.036	0.064
LISA	0.077	0.077	0.077
NPL	0.057	0.053	0.055

Cuadro 1: Resultados obtenidos en las distintas colecciones de prueba

Los bajos resultados obtenidos fueron unas de las razones por la que se realizaron variaciones en el preprocesamiento textual realizado. Mejoras significativas fueron alcanzadas cuando se decidió no realizar *stemming*. Los resultados obtenidos se muestran en el Cuadro 2.

Corpus	**Precisión**	**Recobrado**	**F1**
Cranfield	0.270	0.270	0.270
CISI	0.172	0.124	0.137
LISA	0.163	0.163	0.163
NPL	0.166	0.154	0.159

Cuadro 2: Resultados obtenidos en las distintas colecciones de prueba sin realizar *stemming*

5. Discusión

Para analizar las ventajas y desventajas de nuestro sistema es necesario analizar, por supuesto, las características del modelo de recuperación utilizado. Entre

las ventajas del modelo vectorial está el esquema de ponderación $tf - idf$ para los documentos mejora el rendimiento de la recuperación. La estrategia de coincidencia parcial permite la recuperación de documentos que se aproximen a los requerimientos de la consulta. Además, la fórmula del coseno ordena los documentos de acuerdo al grado de similitud en la consulta. Por otro lado, una de sus limitaciones es que considera a los términos indexados como independientes. Sabemos que en la realidad sí existe correlación entre algunos. Esta presunción del modelo vectorial aunque pueda parecer una limitación simplifica el proceso de recuperación y en algunos casos mejora su rendimiento. El análisis de correlación de términos requiere de enfoques más avanzados y está sujeto al contexto y la naturaleza de los documentos en términos de variedad temática, por ejemplo.

A pesar de las ventajas del modelo utilizado se cree que es posible la obtención de mejores resultados en las colecciones según las medidas definidas. Para esto es necesario un análisis más profundo de varios de los parámetros del sistema. Además, queda pendiente el anális del rendimiento del mismo con otras medidas de evaluación, principalmente algunas subjetivas.

Por otro lado, la gran variedad de técnicas utilizadas por el sistema de recuperación de información, a pesar de mejorar teóricamente el rendimiento del sistema, provocan lentitud en los procesos de consultas, debido a la gran cantidad de operaciones realizadas. Una análisis sobre la utilidad de estas técnicas es una buena propuesta de trabajo futuro.

Así mismo, en la interfaz visual se presenta determinada lentitud en las consultas realizadas, principalmente en los corpus grandes. Esto, por supuesto, en parte es debido a lentitudes concernientes al sistema, pero la ejecución del mismo en una interfaz más ligera como una consulta realiza estos procesos más rápido. Por lo tanto, la eficiencia de la interfaz usada para un proceso como este que resulta costoso no queda clara.

6. Trabajos futuros

A pesar de la cantidad de funcionalidades implementadas aún quedan varias direcciones de trabajo futuro que pudieran ser implementadas para mejorar el funcionamiento del sistema. Por ejemplo, la implementación de otros modelos de recuperación de información, ya sean clásicos, como el Booleano y el Probabilístico u otros alternativos como el Booleano Extendido, el Vectorial Generalizado, el de Semántica Latente y de Redes Neuronales. Sería interesante comprobar el funcionamiento de cada uno de estos modelos en los distintos corpus analizados.

Además, queda pendiente el uso de otras bases de conocimiento como ontologías o tesauros aparte de WordNet que mejoren la expansión de consultas.

En el sistema se hace uso de varios parámetros, como la constante de suavizado a que permiten amortiguar la variación en los pesos de términos que ocurren poco en la consulta y los valores de α, β, γ usados en el algoritmo de Rocchio. Para estos parámetros son usados los valores más comunes utilizados en la práctica. Cuáles son los mejores valores para estos parámetros en las distintas

colecciones de pruebas usadas es otra dirección de trabajo futuro propuesta para nuestro sistema.

7. Conclusiones

En el presente artículo, se describió la implementación de un Sistema de Recuperación de Información basado en el Modelo de Recuperación de Información Vectorial. Se expone el diseño del sistema según cada etapa de la recuperación de información. Varias de sus funcionalidades son explicadas: cómo se realiza el preprocesamiento y la representación de la consulta y los documentos, las principales características y cómo es implementado el MRI Vectorial, cómo se realiza la retroalimentación en el sistema, la expansión de consultas y el agrupamiento de documentos. A su vez, se expone como son usadas técnicas de *Crawling* para generar más información para el sistema y como es realizada y añadida a nuestro sistema la recomendación de documentos. Las herramientas empleadas para la programación y la estructura de nuestro proyecto es explicado. Además, se describe como se utiliza nuestro sistema y los aspectos más importantes de la interfaz visual del mismo. Luego, se realiza una evaluación del sistema en distintas colecciones de prueba empleando distintas métricas estudiadas en clase. Un análisis crítico de las ventajas y desventajas del sistema desarrollado es realizado y por último se identifican direcciones de trabajo futuro.