Proyecto Final de Sistemas de Recuperación de la Información

Autores

Penélope Seijo Avila Victor Lantigua Cano

Tercer Año de Ciencias de la Computación

Universidad de la Habana

Resumen

Este trabajo forma parte de la consolidación del contenido aprendido en la asignatura Sistemas de Recuperación de la Información. Exponemos el modelo escogido para el desarrollo del motor de búsqueda, el modelo vectorial, se explica el diseño e implementación del código de este modelo probándolo con al menos dos colecciones. Además, se explican las métricas de evaluación de este sistema, así como se muestra los resultados obtenidos al desarrollarlas con las distintas colecciones y se presenta una idea para realizar expansión de consultas a través del algoritmo de agrupamiento K-Means (para el cual nos apoyamos también en el uso del Método del Codo) y con apoyo de retroalimentación por parte del usuario. Se propone también las ventajas y desventajas del modelo utilizado, así como recomendaciones para futuros trabajos.

Abstract

This work is part of the consolidation of the content learned in the subject Information Retrieval Systems. We expose the model chosen for the development of the search engine, the vector model, the design and implementation of the code of this model is explained, testing it with at least two collections. In addition, the evaluation metrics of this system are explained, as well as the results obtained when developing them with the different collections and an idea is presented to carry out query expansion through the K-Means grouping algorithm (for which we rely also in the use of the Elbow Method) and with user feedback support. The advantages and disadvantages of the model used are also proposed, as well as recommendations for future work.

Palabras Clave

Modelo vectorial, Sistema de Recuperación de la Información, Retroalimentación, Métricas de Evaluación, K-Means, Método del Codo.

Índice general

1. Introducción

La Recuperación de Información(RI) es el conjunto de actividades orientadas a facilitar la localización de determinados datos u objetos, y las interrelaciones que estos tienen a su vez con otros. Actualmente existen varias disciplinas vinculadas a esta actividad como la lingüística, la documentación o la informática. En este proceso se accede a una información previamente almacenada, mediante herramientas informáticas que permiten establecer ecuaciones de búsqueda específicas. Dicha información ha debido de ser estructura previamente a su almacenamiento, en lo que conocemos hoy en día como los metadatos.
La evolución lógica de los Sistemas de Recuperación de la Información (SRI) ha sido hacia la web, donde han encontrado una alta aplicación práctica y un aumento del número de usuarios, especialmente en el campo de los directorios y motores de búsqueda. El alto grado de consolidación de la web está siendo favorecido por el vertiginoso abaratamiento de la tecnología informática, por el desarrollo de las telecomunicaciones y por la facilidad de publicación de cualquier documento que un autor considere interesante. Los sistemas de recuperación de la información también han ido evolucionando con el fin de adaptarse a este nuevo entorno, hasta llegar a que se han desarrollado algunos de los sistemas más innovadores, al mismo tiempo que extensos y populares. Esta evolución no es un proceso finalizado, sino más bien un proceso en realización.
El problema de la Recuperación de la Información se puede resumir de manera sencilla como "dada una necesidad de información (consulta + perfil del usuario + ...) y un conjunto de documentos, ordenar los documentos de más a menos relevantes para esa necesidad y presentar un subconjunto de aquellos de mayor relevancia"[1]. En la solución de este problema se identifican dos grandes etapas: la elección de un modelo que permita calcular la relevancia de un documento frente a una consulta y el diseño de algoritmos y estructuras de datos que implementen este modelo de forma eficiente. Precisamente en este trabajo trataremos de llevar a cabo un sencillo Sistema de Recuperación de la Información siguiendo estas pautas y basándonos en los conocimientos adquiridos durante el desarrollo de la asignatura. Este sistema se probará con dos colecciones de datos reconocidas, como son la colección de prueba Cranfield y la colección CISI para poder además comprobar distintas métricas de evalución del sistema.

2. Herramientas empleadas para la programación.

Para el desarrollo de este proyecto hemos utilizado el lenguaje de programación Python versión 3.6.5 tanto en el diseño y desarrollo de la Interfaz de Usuario como en el código de la parte lógica de la aplicación que brinda respuesta propiamente a la implementación del SRI.

[1] Francisco J. Martínez Méndez Recuperación de Información: Modleos , Sistemas y Evaluación

Entre las herramientas del lenguaje a destacar se encuentra los módulos utilizados en la parte lógica como la biblioteca Numpy , NLTK y SKLearn que nos brindan herramientas para el trabajo con colecciones y el procesamiento de datos. También usamos los módulos Math, para el trabajo con funciones matemáticas y matPlotLib para graficar .
Utilizamos además el framework Streamlit, desarrollado en dicho lenguaje, para el despliegue de la aplicación web necesaria para la interacción con el usuario.

Para correr la aplicación principal situarse en la línea de comandos en la dirección donde se encuentra el script principal *app.py* y escribir 'streamlit run app.py'.
Para correr la aplicación de manera de prueba de las colecciones que se usan se puede empezar desde el script principal de la lógica del proyecto con 'python searchEngine.py'.

3. Diseño del Sistema de Recuperación de la Información

Los Modelos de Recuperación de Información (MRI) tratan de calcular el grado en que determinado elemento de información responde a determinada consulta. En general esto se consigue calculando los coeficientes de similitud (Coseno, Phi, etc). Los tres modelos más utilizados, por lo que se les considera como modelos clásicos, y que son considerados como modelos bases para el desarrollo de modelos más avanzados son:

Booleano: se crea un conjunto con los elementos de la consulta y otro con los documentos, y se mide la correspondencia.

Vectorial: en el que la consulta y los términos del documento se representan mediante dos vectores, y se mide el grado en que ambos vectores divergen.

Probabilístico: se calcula la probabilidad en que el documento responde a la consulta. Frecuentemente utiliza retroalimentación.

Como hemos comentado en repetidas ocasiones en clases: "no existe un modelo óptimo que dé solución a todos los problemas". Cada MRI sigue esquemas propios de ponderación y, en algunos casos, de ordenación de los documentos por relevancia, como es el caso del Modelo Vectorial el cual es el elegido en nuestro trabajo para desarrollar nuestro SRI, actualmente uno de los más utilizados, que contará además con la técnica de K-Means para la realización de expansión de Consulta y por medio de una interfaz visual una retroalimentación por parte del usuario.

4. Implementación

4.1. Técnicas de procesamiento textual:

Nota: La implementación en código se encuentra en el archivo *collections-parser.py*

La mayoría del texto y los documentos contienen muchas palabras que son redundantes para la clasificación del texto, como palabras vacías, faltas de ortografía, jergas, etc., en el ámbito de la RI esto se conoce como ruido. En muchos algoritmos, como los métodos de aprendizaje estadístico y probabilístico, el ruido y las características innecesarias pueden afectar negativamente el rendimiento general, por lo tanto, la eliminación de estas características es extremadamente importante, aunque en nuestro modelo en específico(modelo vectorial) el ruido no influye de manera negativa en la recuperación si decidimos realizar algunas técnicas para el procesamiento de los datos como la tokenización y la eliminación de lo que conocemos como stopwords. Para llevar a cabo este proceso nos auxiliamos del modulo NLTK que nos brinda herramientas que permiten el trabajo de manera sencilla.

Tokenización:
La tokenización es el proceso de dividir un flujo de texto en palabras, frases, símbolos o cualquier otro elemento significativo llamado token. El objetivo principal de este paso es extraer palabras individuales en una oración.

Eliminación de stopwords:
Una StopList es una lista de palabras que son consideradas como que no tienen valor en el procedimiento de la recuperación. Usualmente a esta lista pertenecen palabras como los artículos, las preposiciones y los pronombres, los cuales específicamente no aportan ningún valor al sentido informacional del texto. En el proceso de eliminación de Stopwords simplemente se eliminan todos los términos del documento a procesar que pertenecen a esta lista.

La primera etapa se basa en la lectura y parseo de las colecciones de prueba para el almacenamiento de los datos , la tokenización de los mismos utilizando el método de NLTK *reguextokenizer()* y la remoción de stopwords utilizando la *stoplist* de NLTK para el idioma inglés, que es el idioma de las colecciones de prueba a utilizar. Se puede encontrar este código en el método *read-collection()* que parsea el archivo que contiene todos los documentos de la colección y el archivo que contiene todas las consultas de prueba. Para parsear una única consulta introducida por el usuario se tiene el método *read-query()*.

4.2. Representación de los documentos y la consulta

Para la representación de los términos del vocabulario nos basamos en la idea de los índices invertidos. En este caso cada termino apunta a una lista de

documentos en los que aparece, con este se hace más eficiente el uso del espacio del almacenamiento; pero con el desarrollo del trabajo no necesitamos como tal los documentos específicos en los cuales cada término aparecía, aunque si la cantidad de documentos total en los que aparecía, por tanto, decidimos llevar un diccionario con esta última información. Este diccionario se encuentra con el nombre de *documents-terms* el cual utilizaremos más adelante para el desarrollo del MRI.
Para la representación de los documentos y de la consulta creamos una clase *File* que nos permitirá un trabajo más cómodo en el desarrollo de una programación orientada a objetos. En esta clase recogeremos los metadatos del documento *(título, autor, info, work)*, estos atributos serán los mostrados luego en la aplicación visual al usuario para los documentos recuperados que respondan a la consulta hecha; y en pasos posteriores veremos cómo nos permitirá guardar valores necesarios como el vector de pesos del documento, lista *weights*, la frecuencia de ocurrencia de cada término en el documento, se representa con el diccionario *frequency*, la cantidad de veces que se repite el término más frecuente *max-frequecy* y la frecuencia normalizada(tf).

4.3. Funcionamiento del Motor de Búsqueda

Modelo Vectorial Nota: La implementación en código se encuentra en el archivo *searchEngine.py*

Cada documento tiene asociado un vector de términos indexados con la información del peso de cada término, en este modelo es un vector ponderado, el peso es positivo y no binario y si un peso es 0 significa que ese término no aparece en el documento; en nuestro caso este vector ponderado solo contendrá los términos que parecen en el documento y se representa con el ya mencionado diccionario *weights*(tf x idf). La idea principal es obtener un vector de pesos por cada consulta al igual que un vector de peso por cada documento.

DEFINICIÓN DE PESO (MODELO VECTORIAL)

En el modelo vectorial, el peso $w_{i,j}$ asociado al par (t_i, d_j) es positivo y no binario. A su vez, los términos indexados en la consulta están ponderados. Sea $w_{i,q}$ el peso asociado al par (t_i, q), donde $w_{i,q} \geq 0$. Entonces, el vector consulta q se define como $\vec{q} = (w_{1q}, w_{2q}, \ldots, w_{nq})$ donde n es la cantidad total de términos indexados en el sistema. El vector de un documento d_j se representa por $\vec{d_j} = (w_{1j}, w_{2j}, \ldots, w_{nj})$.

$$\vec{q} = (w_{1q}, w_{2q}, \ldots, w_{nq})$$
$$\vec{d_j} = (w_{1j}, w_{2j}, \ldots, w_{nj})$$

El peso del término t_i en el documento d_j está dado por:

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

CÁLCULO DE PESOS EN LA CONSULTA

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

Donde $freq_{i,q}$ es la frecuencia del término t_i en el texto de la consulta q. El término a es de suavizado y permite amortiguar la contribución de la frecuencia del término, toma un valor entre 0 y 1. Los valores más usados son 0.4 y 0.5.

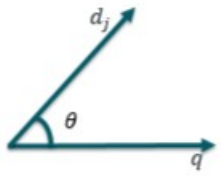
Primero debemos calcular según el modelo la frecuencia normalizada de los términos del documento (tf), medida de similitud intra-documento que permite evitar valores de frecuencia sesgados. Para ello nos guiaremos por la fórmula aprendida utilizando el ya conocido diccionario *frequency* y el valor previamente calculado *max-frequency* para normalizar, guardando el resultado en el parámetro *normalice-frequency* de la clase *File*. Este método lo podemos encontrar en nuestro código en el método *tf()*.
Para la medida de similitud inter-documento (idf) que permite conocer la relevancia de cada término entre todos los documentos, utilizaremos la fórmula con los datos ya conocidos *documents-term*, que muestra el número de documentos en los que aparece cada término y con el valor del total de documentos. Se puede apreciar el procedimiento en el método *idf()* del código.
Luego las fórmulas para el cálculo del peso de cada término presente en el documento y en la consulta difieren, pero ambas usan los datos antes calculados, por tanto se pasa a calcular cada uno según la fórmula correspondiente en los métodos *document-weight()* y *query-weigh()*, usando este último un valor de suavizado alpha con valor por defecto 0.5, el cual permite amortiguar la variación de los pesos de los términos que ocurren poco.
Se devuelve entonces un ranking de los documentos de acuerdo con su grado de similitud a la consulta, utilizando los vectores de pesos antes hallados.

FUNCIÓN DE RANKING

La correlación se calcula utilizando el coseno del ángulo comprendido entre los vectores documentos d_j y la consulta q.

$$sim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \cdot |\vec{q}|}$$

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



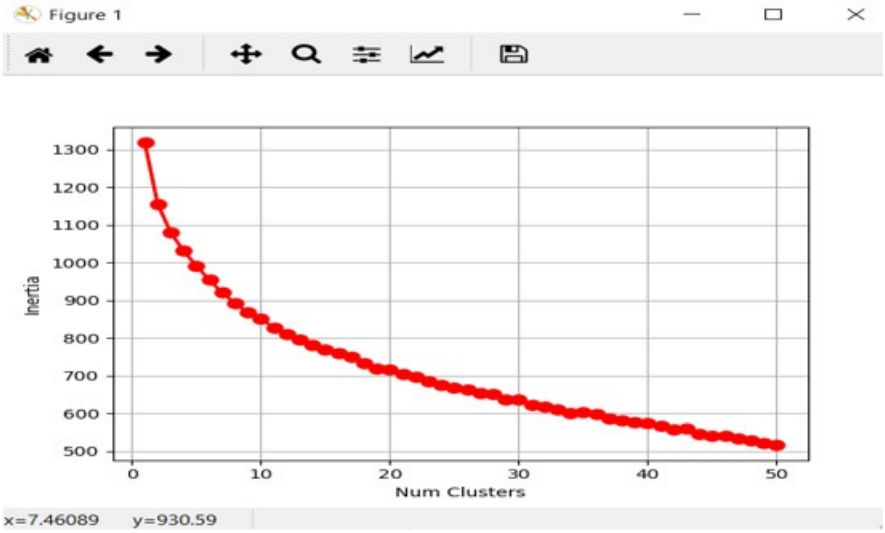
$|\vec{d_j}|, |\vec{q}|$ normas de los vectores documento y consulta respectivamente

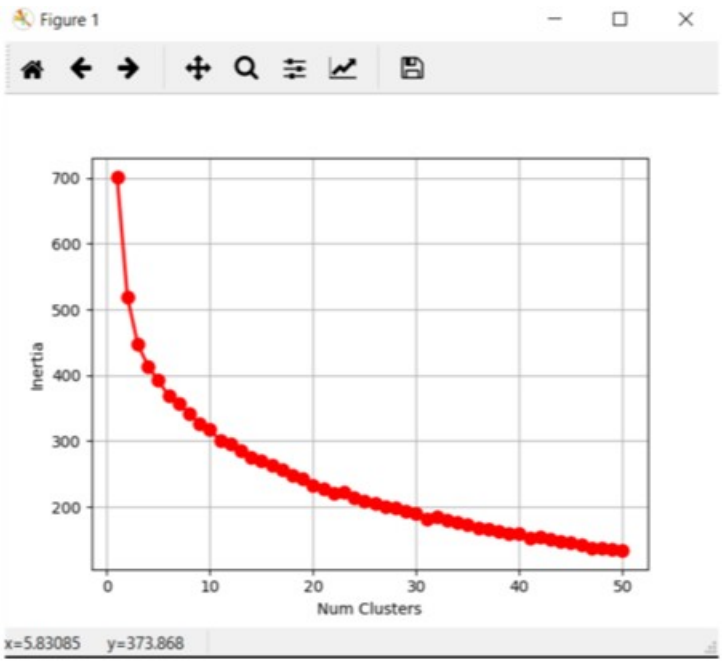
Este valor del ranking se encontrará entre -1 y 1 debido a que se calcula la correlación de los vectores mediante el algoritmo del coseno, en nuestro trabajo se puede apreciar este algortimo en el método *similarity-cos()*. Luego para recuperar los documentos establecemos un umbral y recuperamos aquellos documentos cuyo grado de similitud es mayor que este umbral, se puede apreciar esta etapa en el método *relevance-func()*, en el cual establecimos una función de relevancia donde se clasifican los documentos con un rango del 1 al 5 , mientras menor el número del rango mayor significancia tendrá el documento para la consulta, siendo el valor 5 representante de un documento no relevante.
Aunque el cálculo del peso tf -idf mediante el modelo vectorial intenta superar el problema de los términos comunes en el documento, todavía sufre de algunas otras limitaciones descriptivas. Es decir, tf-idf no puede dar cuenta de la similitud entre las palabras del documento, ya que cada palabra se presenta como un índice(asume que los términos son independientes); además no captura el significado en el texto.
En los últimos años, con el desarrollo de modelos más complejos, como las redes neuronales, se han presentado nuevos métodos que pueden incorporar conceptos, como la similitud de palabras y el etiquetado de partes del discurso. Dos de los métodos más comunes que se han utilizado con éxito para las técnicas de aprendizaje profundo pueden ser word2vec y Glove.
A pesar de sus limitaciones escogimos este MRI debido a que es fácil de implementar y muy usado en la actualidad pues posee técnicas básica para extraer los términos más descriptivos de un documento sencillas.

4.4. Operaciones más avanzadas

Retroalimentación y Expansión de Consultas Para la implementación de la retroalimentación y la expansión de consulta nos guiamos por la idea que

para el modelo vectorial la retroalimentación se puede abordar como un problema de clustering debido a que los vectores de los documentos relevantes para una consulta son similares y tienen una diferencia notable con los vectores de los documentos no relevantes, por tanto llevamos a cabo una retroalimentación donde el usuario escoge de los documentos devueltos por el MRI el que crea más relevante y que mejor se haya adaptado a su interés , y con este realizamos una técnica de agrupamiento mediante el algoritmo K-Means, los documentos que pertenezcan al mismo grupo que el documento relevante seleccionado serán los que se devuelvan como posibles para la expansión de consulta, utilizamos 300 documentos con el objetivo de no volver demasiado lento el algoritmo. En el caso de la prueba con todas las consultas de prueba de la colección realizamos una pseudo-retroalimentación, es decir escogimos nosotros mismos el primer documento del ranking para que sirviera de base.
Para realizar el agrupamiento con el algoritmo K-Means utilizamos la herramienta que nos proporciona el módulo SKLearn*k-means()*. Uno de los problemas que nos encontramos a la hora de aplicar alguno de los métodos de clustering es la elección del número de clústeres. No existe un criterio objetivo ni ampliamente válido para la elección de un número óptimo de clústeres, aunque si se han implementado diferentes métodos que nos ayudan a elegir un número apropiado de clústeres para agrupar los datos, como es el Método del Codo.
El Método del Codo utiliza los valores de la inercia obtenidos tras aplicar el K-Means a diferente número de clústeres, siendo la inercia la suma de las distancias al cuadrado de cada objeto del clúster a su centroide. Una vez obtenidos los valores de la inercia representamos en una gráfica lineal la inercia respecto del número de clústeres. El punto en el que se observa un cambio en la inercia nos dirá el número óptimo de clústeres a seleccionar. El algoritmo para llevar a cabo este procedimiento lo podemos encontrar en el archivo*elbowMethod.py*.
A continuación, mostramos las gráficas obtenidas para la colección Cranfield y CISI .



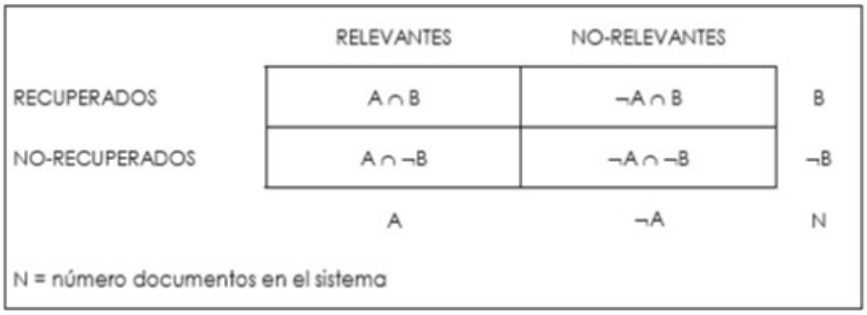


En estos casos el punto del codo no se observa tan evidente como en otros sets de datos, pero al analizar la gráfica decidimos escoger el valor 7 para Cranfield y 5 para CISI pues a partir de ese valor es que evidenciamos un cambio en los valores de la línea de inercia. También podríamos pensar en escoger el valor 2, que es donde, sin lugar a dudas, hay un cambio significativo, y esto tiene todo el sentido del mundo porque estaríamos clasificando directamente los documentos en relevantes y no relevantes con respecto a una consulta, pero con fines de realizar una expansión y una clasificación un poco más profunda escogimos los valores antes mencionados para cada colección como número de clúster para nuestro algoritmo K-Means.

5. Evaluación del Sistema

Nota: La implementación en código se encuentra en el archivo *evaluationSystem.py*
Los SRI, como cualquier otro sistema, son susceptibles de ser sometidos a evaluación, con el fin de que sus usuarios se encuentren en condiciones de valorar su efectividad y, de este modo, adquieran confianza en los mismos.
Primeramente para evaluar nuestro sistema clasificamos los documentos según su relevancia o no y teniendo en cuenta si fueron recuperados o no por el MRI, comparando con los documentos respuesta, documentos que se parsean y analizan en el método *read-collection-answers()*, que se proporcionan en la colección

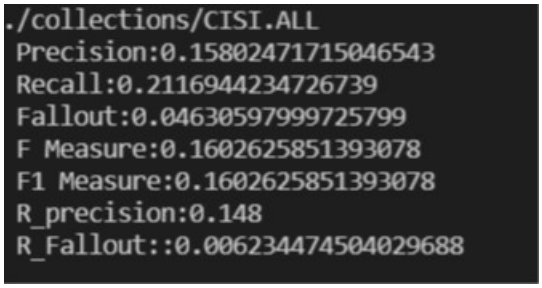
de prueba, esto se puede apreciar en el método *classification()*. A continuación, se muestra una imagen de cómo sería la clasificación.



Evaluamos el sistema empleando las métricas estudiadas en conferencia Precisión, Recobrado(recall), Medida F, Medida F1 y Tasa de Fallo(fallout), para tener en cuenta los documentos irrelevantes también. Además, como estas medidas están diseñadas para modelos q no hacen ranking de los documentos respuesta, y en este modelo si tenemos este ranking entonces analizamos una cantidad fija de documentos dada la ordenación, por ejemplo, los n de mayor valor de relevancia dando lugar a r-medidas, en este caso utilizamos la R-Precisión y el R-Fallout.
Se muestra con un umbral prefijado que comenzamos a trabajar los valores de las métricas que fueron devueltos para la colección CISI, cuyos valores son muy similares a los recuperados con la colección Cranfield.

```
./collections/CISI.ALL
 Precision:0.3064419306184012
 Recall:0.04032079334669205
 Fallout:0.002787962218808017
 F Measure:0.0623854127299191
 F1 Measure:0.0623854127299191
 R_precision:0.10800000000000001
 R_Fallout::0.002388948062382537
```

En este ejemplo se observa que es muy mala la recuperación de documentos pues recupera muy pocos y esto es provocado porque se usa un umbral superior en la mayoría de valores que devuelve la función de similitud, en búsqueda de mejores resultados decidimos seguir probando más con cada colección y buscar el valor del umbral que mejor se ajuste a la colecciones, hasta que obtuvimos con los valores actuales que se encuentran en código en el método *relevance-func()* del archivo *searchEngine.py* los siguientes resultados, que se muestran son los mejores obtenidos.



6. Análisis Critico de las Ventajas y Desventajas del Sistema Desarrollado

Además de las ya planteadas anteriormente podemos considerar:

Ventajas

- Fácil de implementar y de entender
- El modelo vectorial es muy versátil y eficiente a la hora de generar rankings de precisión en colecciones de gran tamaño, lo que lo hace idóneo para determinar la equiparación parcial de los documentos.
- 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.

Desventajas

- Necesita de la intersección de los términos de la consulta con los documentos, en caso contrario no se produce la recuperación de información.
- Al ser un modelo estadístico-matemático, no tiene en cuenta la estructura sintáctico semántica del lenguaje natural.
- Asume que los términos indexados son mutuamente independientes.
- Si la colección tiene un solo documento las formulas son erróneas porque el log se indefine al ser N=1 y =1.
- Establecer un umbral para definir qué documentos son relevantes y cuales no es un valor que no es fijo y fiable a cada colección o sea para cada colección hay que probar y reajustar para ver que umbral se ajusta mejor y permite tener métricas de evaluación más compensadas.
- En general recupera documentos cuando hay igualdad de palabras entre el documento y la consulta no tiene en cuenta sinónimos.

7. Recomendaciones para trabajos futuros que mejoren la propuesta

Entre las desventajas que se mencionaron se encontraba que asume que los términos son independientes, se podría resolver esto usando tesauros, o diccionarios de temáticas. Para la retroalimentación utilizando la interfaz visual que hemos creado, además de pedir al usuario un documento relevante se podría pedir que este diera feedback a la aplicación después de plantear la consulta marcando varios documentos que considera relevantes y cuáles no, que permitirían aplicar el algoritmo de Rocchio estudiado en clases.

8. Interfaz Visual

Para el desarrollo de la interfaz visual como ya comentamos utilizamos el framework Streamlit, debido a que posee una curva de aprendizaje excepcional que reduce el tiempo de aprendizaje, permite el desarrollo puro en Python ya que no necesita de conocimientos de HTML o CSS y además permite un rápido despliegue de la aplicación con tan solo tener instalado el paquete.
El diseño visual es bien sencillo, casi minimalista, donde básicamente se muestra un input para recoger la consulta hecha por el usuario y un botón para hacer la búsqueda. Además en la parte izquierda se permite escoger la colección de prueba con la que se desea trabajar.
Al terminar se muesran los documentos encontrados para la consulta y un botón para marcar uno de ellos como muy relevante y poder realizar la expansión de la consulta.

9. Conclusiones

A pesar de que las métricas obtenidas no fueron las ideales el desarrollo del SRI nos permitió evidenciar la importancia que tiene el modelo vectorial como modelo sencillo, seguro y eficaz para la Recuperación de la Información, no en vano se utiliza como base para el desarrollo de sistemas más avanzados. Como pudimos evidenciar se mejora la respuesta para el usuario al combinar este modelo con diferentes técnicas, como la expansión de consultas y la retroalimentación, incluso con las técnicas propuestas para futuros trabajo la recuperación sería mucho más cercana a las necesidades del cliente, por supuesto , esto teniendo en cuenta la participación del usuario en el proceso, lo cual sabemos que es un evento inseguro y del cual no podemos siempre confiar al cien por cien pues dependemos de personas no expertas. El rápido crecimiento de la web ha hecho necesario el surgimiento de mayores Sistemas de Recuperación de la Información y el desarrollo de este campo de la Ciencia de la Computación, por eso este trabajo nos ha permitido adentrarnos en este mundo aún de manera un tanto incipiente comparado con el desarrollo del nivel actual, pero sin lugar a dudas sentando bases sólidas en el conocimiento de este campo.

10. Bibliografía

Frakes Ricardo Baeza-Yates William B.(s.f.). Information Retrieval: Data Structures and Algorithms.

Christopher D. Manning, P. R. (2009). An Introduction to Information Retrieval. England: Cambridge UP.

Carlos Fleitas, Conferencias Sistemas de Informacion 3er Año de Licenciatura en Ciencia de la Computación, Departamento de Programación, Facultad de Matemática y Computación Universidad de La Habana.

Ricardo Baeza-Yates, Modern Information Retrieval

Francisco Javier Martínez Méndez Recuperación de Información: Modleos , Sistemas y Evaluación

sitio oficial de SKLearn `https://scikit-learn.org`

sitio oficial de Streamlit `https://streamlit.io/`

sitio oficial de NLTK `https://www.nltk.org/`