PROYECTO FINAL DE SISTEMA DE INFORMACIÓN

Carlos Aryam Martínez Molina

UNIVERSIDAD DE LA HABANA, 2021

Contenido

Resumen

Este trabajo forma parte del aprendizaje y consolidación del contenido de Sistema de Recuperación de Información, se da una breve reseña de los modelos clásicos y se explica en profundidad uno de ellos, 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 y se da una explicación de lo que es la retroalimentación y como se utilizaría en mi modelo. Se propone también las ventajas y desventajas de este así como recomendaciones para futuros trabajos.

Palabras Claves

Sistema, Recuperación, Información, Retroalimentación, Vectorial, Booleano, Probabilístico, Relevancia, Evaluación

Introducción

El papel que la investigación tiene en el desarrollo de una ciencia evidente. Naturalmente, la recuperación de información (RI) no es una excepción, y su desarrollo progresivo como disciplina ha venido marcada, para lo bueno y para lo malo, por los resultados de la investigación realizada.

La recuperación de información es un campo donde se realiza una actividad importante de práctica profesional y, también, de investigación científica. Por un lado, dio lugar a la aparición de la industria de la información, con sus bases de datos y sistemas de información, con un marcado componente práctico y profesional centrado en el uso de las bases de datos para satisfacer las necesidades de información de los usuarios. Por otro lado, la investigación científica ha estado dirigida hacia el diseño de sistemas de recuperación de información (SRI) más eficaces, dando lugar a diversas teorías, modelos y experimentos en los que la evaluación ha ocupado un papel central. A lo largo de la historia no ha existido una relación muy estrecha entre ambos componentes, científico y profesional, al menos hasta en los últimos años en los que parece haberse producido un cierto acercamiento y una implementación de las teorías y modelos experimentales en las aplicaciones comerciales.

La investigación en recuperación de información ha favorecido sobre todo la estimulación de ideas; estas ideas, conforme se han ido explorando y comprobando en sistemas experimentales, se han convertido en teorías o modelos. Por lo tanto, estas ideas, teorías y modelos desarrollados han estado influenciados en gran parte por la comprensión y el conocimiento empírico obtenido a partir de experimentos en los que la evaluación ha estado omnipresente. Robertson reconoce que no hay una teoría general y global de recuperación de información, y puede que nunca la haya, aunque el descubrimiento de nuevos modelos y teorías contribuirán a una mejor comprensión.

Son muchos los experimentos, tests e investigaciones llevadas a cabo en el campo de la recuperación de información, y también muchas las críticas y discusiones sobre los resultados obtenidos derivadas, en su mayoría, por la diferente metodología utilizada, en parte debida a la gran variedad de problemas investigados y, en parte, debida también a la diversa procedencia académica de los investigadores. Ole Pors señala que esto último es a la vez una fortaleza y una debilidad. Una fortaleza, porque supone un alto grado de importación de enfoques y teorías interesantes de otras disciplinas académicas. Una debilidad, porque hace muy difícil el desarrollo científico en el sentido de acumulación de teorías y conocimiento. El autor destaca la carencia de un lenguaje científico común que defina bien los conceptos fundamentales, algo que caracteriza a una disciplina científica madura, siendo uno de los principales problemas que caracterizan los experimentos en recuperación de la información.

Desarrollo

Diseño del Sistema de Recuperación de Información

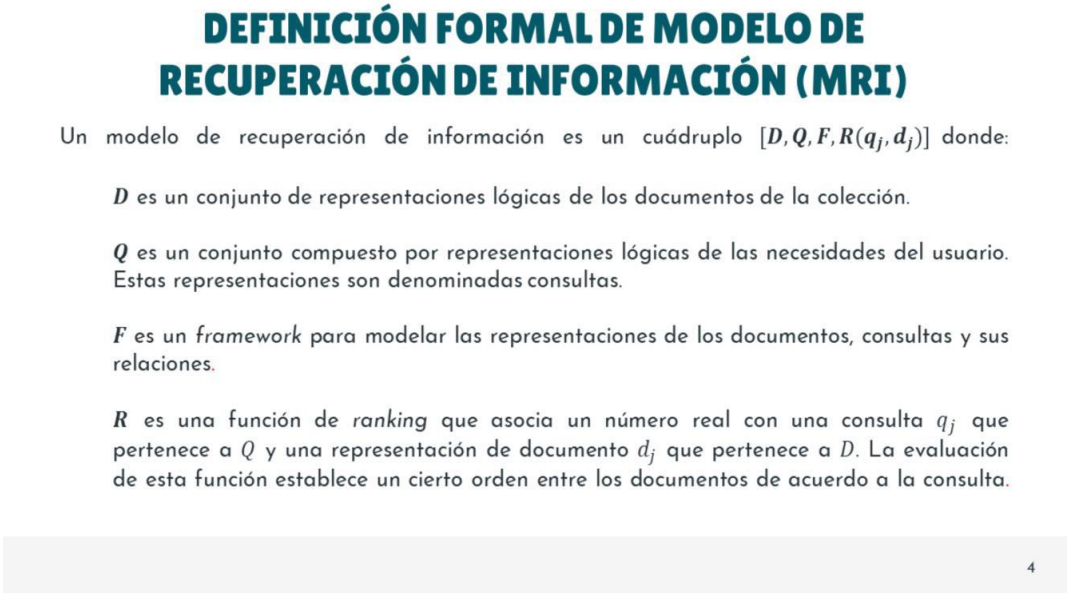
Existen tres modelos clásicos en el estudio de la recuperación de información estos son el booleano, el vectorial y el probabilístico los cuales son denominados como los modelos base para otros modelos más avanzados. Es necesario aclarar que no existe un modelo óptimo que dé solución a todos los problemas.

A grandes rasgos para no adentrarnos en cada modelo porque ya se vio en las conferencias de la asignatura podemos decir que cada MRI sigue esquemas propios de ponderación y, en algunos casos, de ordenación de los documentos por relevancia. Aunque el modelo booleano no lo hace porque este establece una correspondencia exacta con los documentos, simplemente establece si el documento es relevante (1) o no (0).

En el modelo probabilístico los pesos de los términos indexados son binarios. Sin embargo, debido a las características de la función de *ranking*, el modelo permite obtener una ordenación de los documentos. La similitud entre la consulta y un documento está definida como el cociente de dos probabilidades, la probabilidad de que el documento sea relevante dividida por la probabilidad de que no sea relevante, ambas respecto a una consulta q. Esta función de similitud expresa la relación entre la relevancia o no relevancia de un documento frente a una consulta.

Por último se profundizará en la explicación del modelo vectorial más adelante debido a que fue el seleccionado para su implementación. Se escogió porque después de haber estudiado, analizado y comprendido los tres modelos clásicos se llegó a la conclusión, que este era el más indicado por ser el mayor balanceado en cuanto a dificultad, o sea el booleano parece ser un poco sencillo a la vez que el probabilístico se complejiza un poco, por eso se considera que el vectorial es adecuado. Este permite enfocarse más en el proceso de recuperación de información en sí que una implementación que puede llegar a complicarse por cálculos probabilísticos que pueden llegar a ser engorrosos. Con esta opción queda más limpia la implementación y aprendizaje del motor de búsqueda que es el objetivo del proyecto.

Diseño completo del sistema según cada etapa de la recuperación de información



DEFINICIÓN FORMAL DE MODELO DE RECUPERACIÓN DE INFORMACIÓN (MRI)

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

$\boldsymbol{D}$ es un conjunto de representaciones lógicas de los documentos de la colección.

$\boldsymbol{Q}$ es un conjunto compuesto por representaciones lógicas de las necesidades del usuario. Estas representaciones son denominadas consultas.

$\boldsymbol{F}$ es un *framework* para modelar las representaciones de los documentos, consultas y sus relaciones.

$\boldsymbol{R}$ es una función de *ranking* que asocia un número real con una consulta q_j que pertenece a Q y una representación de documento d_j que pertenece a D. La evaluación de esta función establece un cierto orden entre los documentos de acuerdo a la consulta.

4

Imagen 1: Definición formal de MRI

La definición formal del MRI nos va a servir de guía para explicar cada paso del funcionamiento del modelo escogido.

El modelo clásico vectorial en lugar de predecir si un documento es pertinente o no, da un ranking de los documentos de acuerdo con su grado de similitud a la consulta. Luego para recuperar los documentos establecemos un umbral y recuperamos aquellos documentos cuyo grado de similitud es mayor que este umbral. En este modelo el peso está definido como un valor positivo y no binario. A partir de este punto voy a dar un recorrido por el procedimiento, a la vez que la implementación.

La idea principal es obtener un vector de pesos por cada consulta al igual que un vector de peso por cada documento, entonces la componente i-esima del vector es el peso del término i en el documento j de igual manera se obtiene el vector de las consultas y ahora es el término i con la consulta j.

El primer paso es parsear la colección, por cada documento se va a guardar cada palabra que van a ser los términos, en la implementación esto está representado por una lista de tuplas de un valor numérico indicando el id del documento y un diccionario que tiene el término como llave y la cantidad de veces que aparece en ese documento como valor. En ese mismo método además se guarda en otro diccionario, cada término de la colección como llave y la cantidad de documentos en los que aparece como valor (después va a ser necesario).

Luego se calcula para cada documento su vector de pesos, primero obteniendo la frecuencia normalizada ($tf_{i,j}$) que resulta en un vector donde cada componente es la cantidad de veces que aparece un término dividido por el valor del término que más se repite, Esta es una medida de similitud intra-documento. Se considera la frecuencia de cada término en un documento dividido entre la frecuencia máxima en ese mismo documento para normalizar esta medida. Esto se hace con el objetivo de evitar valores de frecuencia sesgados por la longitud del documento. Si a un documento añadiéramos una copia de él mismo tendríamos que las frecuencias de ocurrencia de los términos en el nuevo documento serían el doble de las anteriores. Sin embargo, el documento duplicado no es más relevante que el original.

Luego se calcula la frecuencia de ocurrencia de un término t_i dentro de todos los documentos de la colección esto se llama idf del inglés inverse document frequency. Por cada documento se realiza el siguiente procedimiento, se recorre cada término del documento y se aplica la fórmula $idf_i = \log\frac{N}{n_i}$ donde N es la cantidad de documentos de la colección y n_i la cantidad de

documentos en los que aparece el término t_i . Además de la medida de la 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. Es por esto que se introduce el inverso de la frecuencia de un término en la colección de documentos. Esta es una medida de similitud inter-documentos.

Después de obtener el $tf_{i,j}$ y el idf_i por cada documento se forma el vector de pesos de este, multiplicando estas dos cada componente del vector $tf_{i,j}$ por el escalar idf_i. De manera análoga se obtiene el vector de pesos para las consultas el $tf_{i,j}$ y el idf_i se calculan igual solo que ahora el conjunto de documentos es un conjunto de consultas por lo que se realiza el mismo proceso para obtenerlos solo que a la hora de formar el vector de pesos en las consultas en vez de utilizar la fórmula

$w_{i,j}$= $tf_{i,j}$ × idf_i se usa $w_{i,q}$= (a + (1 - a) $tf_{i,q}$) × idf_i.

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. Una vez que se tienen los vectores de pesos formados solo queda aplicar la función de similitud que se realiza de cada consulta con cada documento devolviendo un valor entre -1 y 1 porque la fórmula está dada por el coseno del ángulo y se utiliza la imagen de conferencia para ilustrar la fórmula que da vida a la función de ranking.

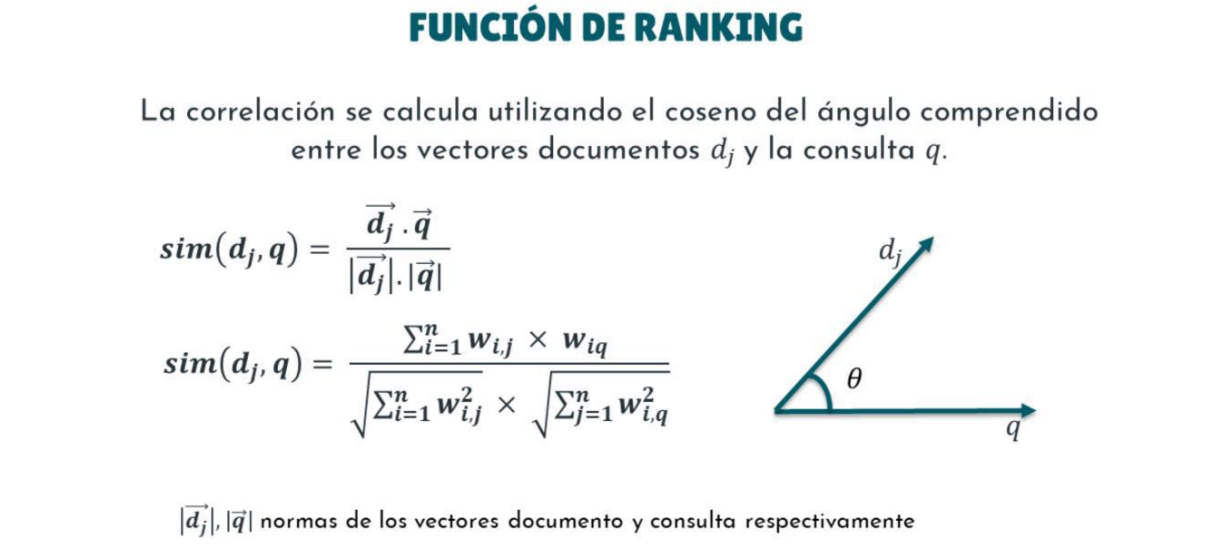


Imagen 2: Función de ranking

Una vez aplicado este proceso, cada consulta le da un valor de similitud a cada documento, solo queda ordenar esos valores y establecer el umbral a partir del cual se va a tomar como relevante el documento. Por último se puede tener una función de relevancia e ir dando valores entre 5 y 1 a intervalos del valor dado por la función similitud. Por ejemplo se le asigna valor 5 a los documentos y consultas que tengan una similitud <= que 0 y se toma como que 5 es no relevante y del 4 al uno si lo es siendo 1 una perfecta similitud donde de 1 el coseno del ángulo que indicaría que la consulta es igual que el documento.

Presentación de las herramientas empleadas para la programación y aspectos más importantes del código.

Herramientas empleadas

Se utilizó como lenguaje de programación Python versión 3.6.4

Se importó la librería math de Python para la utilización de fórmulas como la raíz cuadrada y el logaritmo.

El proyecto va a quedar archivado en github y se puede acceder mediante el enlace este

Aspectos importantes del código

El código está conformado por tres archivos .py el principal y por el cual hay que arrancar el proyecto es el vectorialModel.py. Para correrlo hay que hacer "Python vectorialModel.py" asegurarse antes de haber colocado en las variables url del método "main" las direcciones de los archivos que quiera probar, si desea cambiar el que se usa por defecto en el proyecto.

En este archivo es donde se realiza el proceso del motor de búsqueda donde se calcula el $tf_{i,j}$ y el idf_i además de calcular el ranking en la función "similitud" y ordenarlos en la función "rank". Los métodos "documentWeight" y el "queryWeight" son los que devuelven los vectores de los pesos después de haberles realizado todo el proceso y haber aplicado las fórmulas de cada uno, para eso se reutiliza en ambos los métodos "matrix_tf" y "weightVector_idf" que dan el valor de el $tf_{i,j}$ y el idf_i respectivamente, luego en "makeMatrix" se aplica el cálculo, por eso esta función recibe un lambda que especifica que fórmula usar porque es usada por ambos por las query y los documentos para reutilizar código.En específico en la función de las "query" llamada "queryWeight" se recibe un parámetro que toma por defecto 0.5 si no se le pasa ninguno que es el valor de suavizado a.

Para realizar todo el proceso anterior es necesario tener los documentos o consultas parseados por eso en el otro archivo utils.py está la función "readDocument" que es la que parsea el documento o consulta y devuelve una lista de tuplas que tienen el id del documento como diccionario que tiene como llave cada palabra del documento y como valor las veces que se repite, además de otro diccionario que tiene palabras de toda la colección y la cantidad de documentos en los que se repite. Este método se apoya en varios otros en el mismo archivo utils.py. Por último en el archivo evaluationSystem.py es donde se calculan las métricas con cada variable que necesite como son la cantidad de documentos Recuperados Relevantes y los demás explicados en la conferencia 4.

R-Precisión

La R-Precisión forma parte de las métricas de evaluación y consiste en aplicar la fórmula de Precisión a partir de la posición R del ranking.

En mi solución no se utiliza R-Precisión porque los documentos relevantes para mi están dados a partir de que sus valores sean mayores que el umbral establecido.
Para agregar la R-Precisión a mi solución basta con establecer un valor R y tomar solo como relevantes los primeros R documentos del ranking. De manera análoga se puede aplicar este criterio al recobrado y a la medida_f.

Retroalimentación en el MRI clásico Vectorial

Surge mediante la limitante de que muchas veces los MRI no logran dar respuesta a la necesidad de información del usuario, debido a la complejidad de la misma y a la manera en que se expresan como consultas. Por esto una idea interesante podría ser involucrar al usuario en un proceso de retroalimentación que permita ir perfeccionando la respuesta a las consultas.

De forma genérica los pasos para la retroalimentación son:

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 utilizando esta información
5. Jump a 2

Algoritmo de Rocchio

El llamado *algoritmo de Rocchio* [24] es bien conocido y aplicado en la realimentación de consultas. En este ámbito, la idea es simple: formulada y ejecutada una primera consulta, el usuario examina los documentos devueltos y determina cuáles le resultan relevantes y cuáles no. Con estos datos, el sistema genera automáticamente una nueva consulta, basándose en los documentos que el usuario señaló como relevantes o no relevantes. En este contexto, el algoritmo de Rocchio proporciona un sistema para construir el vector de la nueva consulta, recalculando los pesos de los términos de ésta y aplicando un coeficiente a los pesos de la consulta inicial, otro a los de los documentos relevantes y otro distinto a los de los no relevantes.

El Standard Rocchio es un algoritmo que permite reformular una consulta en un Sistema de Recuperación de Información (SRI) que utiliza el modelo vectorial, por lo que trabaja sobre vectores. Se logra que se incremente la importancia de documentos considerados relevantes (y por tanto se incluirían documentos similares a ellos) mientras que se disminuye la relevancia para aquellos documentos que se hayan definidos negativamente como no relevantes. La fórmula es la siguiente:

q_m = q + $\frac{1}{|D_r|} \sum_{\forall d_j \in D_r} d_j$ - $\frac{1}{|D_n|} \sum_{\forall d_j \in D_n} d_j$

• Se toma q_m como la consulta resultante con retroalimentación incluida.
• q es la consulta original.
• D_r es el conjunto de documentos relevantes devueltos para la consulta.
• $D_n$es el conjunto de documentos no relevantes devueltos para la consulta.
• d_j es el vector documento.

Como ejemplo suponga tres documentos d1, d2 y d3 en la colección. En el vocabulario de la colección hay tres palabras distintas por lo que cada vector tiene (A, B, C).

d_1= (0.18, 0.18, 0.18)
d_2= (0.18, 0.18, 0)
d_3= (0, 0, 0.18)

La consulta por parte de un usuario tiene la siguiente forma:

q = (0.18, 0.18, 0.09)
Por lo que obtenemos la similitud entre los documentos:

• sim (d_1, q) = 0.96
• sim (d_2, q) = 0.96
• sim (d_3, q) = 0.33

A partir del ejemplo anterior, se sabe debido a la retroalimentación que los documentos d_1 y d_2 son relevantes y que el documento d_3 **no** es relevante, entonces aplicamos la fórmula:

q_m = q + $\frac{1}{2}(d_1 + d_2)$ - $\frac{1}{1}(d_3)$ = (0.36, 0.36, 0)

Ahora, con la consulta reformulada obtenemos una nueva similitud entre los vectores:

- sim(d_1, q) = 0.82
- sim(d_2, q) = 0.97
- sim(d_3, q) = 0

Ahora como incorporaría la retroalimentación a mi modelo vectorial ya realizado?
El proceso de obtención de los pesos de los documentos y las consultas se mantendrían al igual que el cálculo de la fórmula de similitud, pero ahora se le agregaría una interfaz visual para intercambiar con el usuario y con la que este podría dar feedback a la aplicación después de plantear la consulta marcando que documentos considera relevantes y cuáles no, que son los valores D_r y D_n que me permitirían aplicar el algoritmo de Rocchio y obtener el Nuevo vector consulta así como los nuevos resultados de similitud entre los documentos y estos nuevos pesos del vector consulta, volver a ordenar los documentos por el valor de similitud darle el resultado al usuario y comenzar el proceso otra vez hasta que el cliente este satisfecho con el resultado obtenido.

Evaluación del sistema empleando las métricas

Ranking de las primeras 5 consultas de la colección Cranfield.

1 13 3
1 12 4
1 51 4
1 184 4
2 12 2

2 51 3
2 100 4
2 253 4
2 429 4
3 144 2
3 399 2
3 5 3
3 181 3
3 90 4
3 119 4
4 166 3
4 185 4
5 103 3

Métricas de evaluación.

Precisión: 0.1701170117011701
Recobrado: 0.10288513881328253
Medida_F (beta=1): 0.1282225237449118

Ranking de todas las consultas de la colección CISI para el mismo umbral que Cranfield.

9 1120 4
10 1108 4
10 1385 4
34 1180 3
62 430 4
72 10 4
72 72 4
92 4 4
92 148 4

92 528 4

Métricas de evaluación.

Precisión: 0.3
Recobrado: 0.0009633911368015414
Medida_F (beta=1): 0.0019206145966709348

En este ejemplo se ve que es muy mala la recuperación de documentos pues recupera muy pocos y esto es provocado porque se usa un umbral superior a la mayoría de valores que devuelve la función de similitud esto nos enseña y empuja a testear más con cada colección y buscar el valor del umbral que mejor se ajuste a la colección en cuestión.

Ranking de las 2 primeras consultas de la colección CISI para un umbral que mejora las métricas.

1 17 3
1 1245 3
1 8 4
1 12 4
1 236 4
1 323 4
1 934 4
1 1165 4
1 1206 4
1 1323 4
2 166 3
2 17 4
2 1399 4

Métricas de evaluación.

Precisión: 0.08249852681202122
Recobrado: 0.0449582530507386
Medida_F (beta=1): 0.05819995842860112

Luego de probar con varios umbrales el que mejor resultados daba fue este, no es aun bueno en cuanto a las métricas pero mejoró bastante.

Ventajas y desventajas del sistema desarrollado

Ventajas

- 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 fórmula del coseno ordena los documentos de acuerdo al grado de similitud con la consulta.
- 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 $\log\frac{N}{n_i}$ se indefine al ser N=1 y n_i=1.

- Establecer un umbral para definir que 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.

Conclusiones

La implementación del modelo vectorial de Recuperación de Información dio resultados positivos, la sencillez del modelo hace que su eficacia pruebe la fortaleza de este algoritmo. Aunque se evidencia que este modelo trabaja efectivamente para colecciones de tamaño razonable, en el mundo cambiante y con cada vez más información para procesar es necesario seguir mejorando el proceso de recuperación de información, si con la retroalimentación existe una mejora en el rendimiento, de algo podemos estar seguros es que en lo adelante necesitaremos algoritmos más poderosos y muchas otras mejoras en estos mismos modelos para estar a la altura de la creciente era de la información.

Recomendaciones para trabajos futuros

Incorporar retroalimentación, expansión de consultas, integración con algoritmos de Crawling, operaciones textuales más avanzadas, uso de bases de conocimientos como ontologías.

Como recomendación interesante se propone agregar como feature la posibilidad de ajustar automáticamente el umbral para la colección en cuestión realizar un proceso que seguro va a ser lento para probar que umbral se ajusta más. Es una bonita aplicación en la que se pueden emplear algoritmos de búsqueda binaria para reducir el costo temporal pero ayudaría mucho al momento de probar muchas colecciones.

Bibliografía

Aparicio, C. F. (2021). Conferencias de Sistemas de Información. La Habana.

Baeza-Yates, W. B. (s.f.). *Information Retrieval: Data Structures & Algorithms.*

Carlos G. Figuerola, J. L. (2004). Algunas Técnicas de Clasificación Automática de Documentos. *Cuaderno de documentación multimedia*.

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

Salvador Oliván, J., & Arquero Avilés, R. (s.f.). La investigación en Recuperación de Información: Revisión de tendencias actuales y críticas.