Sistemas de Recuperación de Información

Miguel Alejandro Asin Barthelemy MIGUEL.ASIN@ESTUDIANTES.MATCOM.UH.CU
Grupo C511

Yan Carlos González Blanco YANCARLOGLEZ98@GMAIL.COM
Grupo C511

Henri Estévez Gómez HENRY.ESTEVEZ1999@GMAIL.COM
Grupo C511

Tutor(es):
Lic. Marcel E. Sánchez Aguilar, *Universidad de la Habana*

Resumen

Se presenta el desarrollo de cada una de las partes que componen un modelo de recuperación de información, con la principal finalidad de realizar búsquedas de documentos en una colección de archivos a partir de consultas realizadas por un usuario. Este proceso se descompone en cuatro etapas fundamentales: procesamiento de la consulta, en la cual a partir del modelo definido para la recuperación se representa la consulta como la estructura adecuada para el correcto procesamiento; representación de los documentos, tal como se realizó con la consulta, se realiza también el procesamiento y la representación correspondientes de los documentos; funcionamiento del motor de búsqueda, esto incluye llevar a la práctica el modelo a utilizar, para ello se usó el lenguaje de python gracias a las facilidades que brinda en el procesamiento de lenguaje natural en comparación con otros lenguajes y; la obtención de resultados, en el cual aquellos documentos recuperados son dispuestos ante el usuario a manera de ranking a partir de una puntuación determinada por el nivel de significación de cada documento dada la consulta realizada. Al final se presentará un análisis de los resultados obtenidos por el sistema aplicado junto con recomendaciones para futuras mejoras de implementaciones.

Abstract

The development of each of the parts that make up an information retrieval model is presented, with the main purpose of searching for documents in a collection of files from queries made by a user. This process is broken down into four fundamental stages: processing of the query, in which, based on the model defined for recovery, the query is represented as the appropriate structure for correct processing; representation of the documents, as was done with the query, the corresponding processing and representation of the documents is also carried out; operation of the search engine, this includes putting into practice the model to be used, for which the python language was used thanks to the facilities it provides in natural language processing compared to other languages and; obtaining results, in which those retrieved documents are arranged before the user as a ranking based on a score determined by the level of significance of each document given the query made. At the end, an analysis of the results obtained by the applied system will be presented along with recommendations for future improvements of implementations.

1. Introducción

Desde la antigüedad, desde la Alejandría de Tolomeo, hasta las bibliotecas monásticas de la Edad Media, se definió con claros enunciados, la necesidad del desarrollar métodos y técnicas, que no sólo permitieran el control y conservación de las colecciones, sino también la identificación de cada de uno de sus ejemplares con el objetivo de recuperarlos correctamente. A pesar de que, desde entonces, se utilizaron técnicas para el manejo de los títulos y responder a las escasas necesidades de información de la época, no fue, sino con el desarrollo científico y la especialización, que se maximizó la exigencia sobre estos métodos en aras de satisfacer necesidades cada vez más puntuales.

La invención en 1946 de las tecnologías computacionales fue de progresiva e inmediata aplica-

ción en la naciente esfera de la información, especialmente para solucionar las preocupaciones dominantes en ese lapso de explosión documental, sobre como localizar y buscar información puntualmente[1]. Con un crecimiento cada vez mayor de la cantidad de información surge la necesidad de crear sistemas o modelos que permitieran recuperarla en un tiempo razonable. A lo largo de la historia se han utilizado varias herramientas en el proceso de búsqueda y recuperación y, en el presente artículo se expondrán algunos de ellos.

Cada modelo de recuperación de información se descompone en un conjunto de etapas caracterizadas por la parte del sistema en la que se enfocan: colección de documentos, manipulación de consultas y archivos, modelo computacional y obtención de los resultados. Los modelos a desarrollar estarán compuesto por dichas etapas. Una vez constuidos, serán capaces de, dada una colección de documentos, y una consulta, devolver aquellos relevantes a dicha consulta dispuestos en un orden determinado por su nivel de relevancia (en el caso del modelo que lo permita). Dicho nivel será determinado por el algoritmo en dependencia de las partes de la consultas presentes en el documento.

Una vez construido el sistema, se analizarán los resultados que se obtienen al ejecutarlo, así como posibles recomendaciones para nuevas versiones que mejoren el proceso de recuperación de información.

2. Diseño del sistema

Para el desarrollo del sistema de recuperación, se optó por el uso del modelo vectorial y el modelo booleano, a continuación, se presentará un analisis detallado del empleo de ambos modelos y posteriormente se establecerá una comparación entre ambos atendiendo a los resultados que se obtienen por cada uno en esenarios distintos.

2.1 Modelo vectorial

Para el desarrollo del sistema de recuperación, primeramente se optó por el enfoque brindado por el modelo vectorial, tanto por las facilidades computacionales que brinda respecto al probabilístico, como por la propiedad de presentar los resultados en un sistema de ranking, que permita establecer un orden de relevancia de los documentos encontrados, a partir de la consulta, lo cual es un punto a favor respecto al modelo booleano.

Representación de los documentos y las consultas

El primer aspecto a tener en cuenta es la forma de representar la colección de archivos y la consulta. En el caso de los primeros, se calcula el peso de cada término en los documentos mediante la frecuencia de los mismos (tf) a través de la ecuación:

$$tf_{ij} = \frac{n_{ij}}{\max_k\{n_{kj}\}}$$

donde n_{ij} representa las veces que ocurre el término i en el documento j y, tf_{ij} el valor del tf del término i en el documento j y; el índice de frecuencia invertida (idf), dado por:

$$idf_i = log\left(\frac{N}{n_i}\right)$$

donde N es la cantidad de documentos de la colección y n_i es la cantidad de documentos que contienen al término i; de esta forma cada documento se representa como un vector de pesos con una cantidad de componentes igual a la cantidad de terminos del alfabeto. En el caso de la consulta tf indica la frecuencia de cada término en la misma, mientras que idf mantiene el mismo significado que en el caso de los documentos.

Formalmente para cada documento i se define el peso del término j como w_{ij} dónde:

$$w_{ij} = tf_{ij} \cdot idf_j$$

mientras que en la consulta q:

$$w_{qj} = (\alpha + (1-\alpha) \cdot tf_{qj}) \cdot idf_j$$

donde α representa un valor de suavisado para minimizar la contribución de la frecuencia del término al peso del mismo en la consulta.

Funcionamiento del motor de búsqueda

Dentro del sistema, se realiza un preprocesamiento de texto de los documentos, tras el cual las palabras de los documentos son llevadas a su forma primitiva. Esto favorece la búsqueda de los documentos significativos a una consulta que posee términos relacionados con el documento pero que se encuentran conjugados o expresan un mismo significado pero están escritos de diferente forma.

Una vez se tiene una correcta representación de los documentos y la consulta a procesar, se realiza el proceso de búsqueda. El mismo consiste en el cálculo del coseno del ángulo formado entre los

vectores de la consulta y cada uno de los documentos de la colección:

$$cos(D_i, q) = sim(D_i, q) = \frac{\sum_{j=1}^{n} w_{ij} \cdot w_{qj}}{\sqrt{\sum_{j=1}^{n} w_{ij}^2} \cdot \sqrt{\sum_{j=1}^{n} w_{qj}^2}}$$

El significado geométrico de la fórmula anterior es que, para una consulta dada, a cada documento se le asigna un valor real en dependencia de la cercanía del vector del documento al vector de la consulta, esta cercanía está determinada por el ángulo formado entre ambos vectores, mientras más cercanos sean, menor será el ángulo y mayor el valor del coseno del mismo.

El resultado obtenido para cada documento brinda un valor asociado a la similitud que existe entre ambos vectores (en caso de que el coseno sea cero, ambos vectores son iguales y la similitud sería de 1). El valor obtenido en el rango $[0, 1]$ se asocia a cada documento indicando el nivel de relevancia de cada uno de ellos y se procede a dar el resultado tras realizar un ranking a partir del mismo.

Obtención de los resultados

Tras realizar el proceso de búsqueda de los documentos más relevantes para una consulta dada, se seleccionan aquellos que los sean y un k, que representa el total de documentos del subconjunto de documentos relevantes a mostrar. Note que el algoritmo no clasifica a los documentos en relevantes o no, sino que establece un valor de relevancia. Más adelante se establecerá un umbral que permita clasificar a los documentos en ambos tipos pero, los resultados que se le muestran al usuario se considerarán los mejores candidatos a ser relevantes.

2.2 Modelo booleano

Se presenta un segundo modelo implementado para la recuperación de información: el modelo booleano. En ocaciones el usuario que realiza una consulta puede no estar interesado en los documentos que posean cierta similitud con los términos de su consulta, sino en todos aquellos documentos que posean cada uno de los términos de la consulta al mismo tiempo, incluso, aquellos que solo posean al menos un término en común o simplemente que cumplan con los valores lógicos de los términos de la consulta tal que la misma evalúe como 1.

Representación de los documentos y las consultas

A diferencia del modelo vectorial, en el modelo booleano los documentos se representas como vectores donde sus componentes solo poseen valor 0 o 1 (1 en caso de que el término correspondiente se encuentre en el documento y 0 en caso contrario).

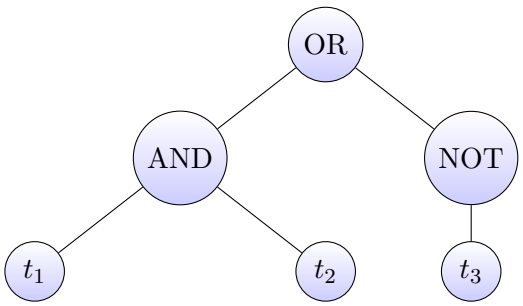
Mientras que la consulta es una fórmula lógica bien formada con operadores AND, OR y NOT, ejemplo:

$$t_1 \ AND \ t_2 \ OR \ NOT \ t_3.$$

Funcionamiento del motor de búsqueda

Tras elaborar una consulta lógica bien formada. Es necesario preprocesar sus términos para poder realizar la búsqueda correspondiente. La forma convencional de recuperar información con este modelo es representar la consulta como una disyunción de conjunciones, y tomar aquellos documentos tales que alguna conjunción coincida con el vector del documento. Sin embargo, debido a la complejidad a la hora de expresar la consulta de esa forma computacionalmente (ver problema de Satisfacibilidad Booleana o SAT, clasificado como NP-Completo), se optó por realizarlo utilizando una estructura de árbol.

Aplicando técnicas estudiadas para el desarrollo de compiladores, se puede aprovechar la estructura de la consulta para crear un árbol de sintaxis y encontrar el valor de la misma. Para ello se realiza un proceso de "tokenización" y "parseo", con el cual se construye dicho árbol en el que cada nodo representa la operación a realizar (teniendo en cuenta la precedencia) y cada hoja constituyen los términos de la consulta. Considere la consulta de ejemplo vista hace un momento, el árbol de sintaxis sería el siguiente:



Este árbol posee la particularidad de que cada nodo representa una operación lógica (AND, OR o NOT) y las hojas, los términos de las consultas.

Para encontrar el valor de una consulta, basta con evaluar el nodo raíz (OR en caso del ejemplo anterior) el cual necesitará evaluar sus nodos hijos

para poder encontrar dicho valor. Este llamado de "evaluar" se realiza en cada nodo de forma recursiva hasta llegar a las hojas.

En el caso de las hojas, tras realizar el proceso de "tokenización", todos los términos se encuentran en su forma primitiva, por lo que el siguiente paso es encontrar, para cada uno, el conjunto de documentos que lo contienen.

Para evaluar los nodos se realiza la operación entre conjuntos correspondiente. En el caso de los nodos AND, el resultado será la intersección de los conjuntos de sus nodos hijos. En el caso de OR, la unión de los mismos, mientras que en los nodos NOT, se toma el complemento del conjunto devuelto por su único hijo, o lo que es lo mismo la diferencia entre el conjunto que contiene todos los documentos y el conjunto devuelto por su nodo hijo.

Por ejemplo, suponga que se tiene una colección de cuatro documentos, tal que los términos t_1, t_2 y t_3 se encuentran en los documentos $\{D_1, D_2, D_3\}$, $\{D_1, D_3, D_4\}$ y $\{D_2, D_3, D_4\}$ respectivamente. Al evaluar el nodo AND, se obtiene el conjunto $\{D_1, D_3\}$, mientras que el nodo NOT, evalúa como $\{D_1\}$. Por último, el nodo OR obtiene como solución el conjunto $\{D_1, D_3\}$, conjunto que contiene los documentos que cumplen con la consulta $t_1\ AND\ t_2\ OR\ NOT\ t_3$.

Obtención de los resultados

Mientras que el modelo vectorial selecciona los mejores k documentos que podrían ser relevantes a una consulta, al usar el modelo booleano, se obtiene un subconjunto de documentos del conjunto inicial, que teóricamente poseen un alto grado de relevancia, sin embargo como no se establece un orden entre ellos, solo se le muestra una parte de estos al ususario.

3. Algoritmo

Para la implementación del sistema, se emplearon varias bibliotecas del lenguaje de programación python, en su mayoría, dedicadas al procesamiento de texto. El por qué del lenguaje: mayor facilidad computacional y, un mayor número de herramientas para el objetivo buscado por el programa.

Siguiendo las partes definidas anteriormente en las cuales está dividido el sistema, se explicará el algoritmo desarrollado.

Representación de los documentos y las consultas

Este aspecto se centra en lo que es el procesamiento de texto y el proceso de indexación de los documentos.

El primer paso del sistema es cargar todos los documentos o páginas que se encuentren en las direcciones especificadas. Dentro del módulo Files.py se realiza este proceso. En el caso de las direcciones locales se carga el contenido de los documentos presentes en la dirección especificada, mientras que en el caso de una dirección URL se establece una conexión con el servidior, se envía la solicitud para la página y una vez recibida se extrae el contenido textual, este se almacena en un archivo local para su posterior análisis.

En un segundo paso se lee el contenido de todos los documentos encontrados, tanto locales como descargados y se realiza el procesamiento de texto. Dicho procesamiento consiste en tomar el texto y realizar el proceso de división en tokens de los términos y la normalización de estos (Token.py) el objetivo de la normalización es reducir cada palabra a su forma primitiva eliminando de esta forma las conjugaciones que puedan dificultar el proceso de recuperación, además de elimitar términos que no aportan información al contenido de los documentos como son las preposiciones, conjunciones, artículos, etc.. Para este paso fue necesario usar la herramienta *nltk* (Natural Language Toolkit, es una suite que contiene bibliotecas y programas para el procesamiento estadístico del lenguaje y una de las bibliotecas de PNL más poderosas, que contiene paquetes para hacer que las máquinas comprendan el lenguaje humano y le respondan con una respuesta adecuada).

Tras extraer la información relevante de cada documento y eliminar las redundancias u otras informaciones innecesarias, el contenido de cada documento habrá sido representado como un arreglo de tokens, donde cada token será una palabra lo mas simplificada posible.

Con estas palabras, en el caso del modelo vectorial, se construye el vocabulario o alfabeto del sistema para luego realizar el cálculo de los valores de tf e idf de los términos y documentos y representar toda la información textual de los archivos y la misma consulta como un vector numérico de los pesos de cada una de sus palabras. Para ello se empleó la clase $TfidfVectorizer$ del módulo $sklearn.feature\_extraction.text$ perteneciente de la biblioteca *sklearn* de python (es una herramienta que permite tomar un conjunto de documentos

y determinar la matriz de valores TF-IDF de los términos de los mismos) ver vectorial.py.

En el caso del modelo booleano, los documentos y las consultas serán representados como un vector de longitud igual a la cantidad de términos del vocabulario y los valores de cada una de las componentes de los vectores de los documentos indicarán si el termino dado se encuentra o no en dicho documento, ver boolean.py.

Funcionamiento del motor de búsqueda

Tras haber realizado el procesamiento del texto de los documentos, se realiza el proceso de búsqueda a partir de una consulta definida por el usuario. Para ello se expresa la consulta como un vector de pesos en el caso del modelo vectorial y como un AST (árbol de sintaxis abstractas) en el caso del modelo booleano.

Para el modelo vectorial, la búsqueda se realiza a partir de los valores obtenidos por la función *cosine_sim* del archivo vectorial.py, mientras que en el modelo booleano, se realiza un proceso de tokenización y parsing y posteriormente se evalúan recursivamente los nodos del AST con el método *get_documents*, ver boolean.py.

Obtención de los resultados

El módulo de búsqueda concluye con el retorno de un arreglo de documentos que representa el conjunto de archivos recuperados a partir de la consulta dada.

En el caso del modelo vectorial, se retorna los k mejores documentos del ranking y un conjunto de posibles sugerencias a partir de la consulta. Estas sugerencias se crean en query_expansion.py. Para ello se usa la biblioteca wordnet (kit de herramientas de *nltk* en python para un procesamiento simple y directo del lenguaje natural). Durante el procesamiento de los documentos, se crea una matriz de correlación de las palabras, donde el valor de una celda (i, j) indica la cantidad de ocurrencias en el conjunto de documentos en los que aparece el término j inmediatamente despúes del término i. Para la expansión de consultas, se realizan dos operaciones usando las herramientas anteriores. Primeramente se seleccionan términos de la consulta y se comprueba en la matriz de correlación si al sustituir el término anterior por uno de sus sinónimos se obtiene un valor en la matriz mayor que cero, en caso positivo, se construye la nueva sugerencia con el sinónimo. La siguiente es usando directamente la matriz de correlación y sugerir una nueva consulta sustituyendo algún término de la misma por otro perteneciente al vocabulario tal que en la matriz de correlación el valor con el nuevo término sea mayor que cero.

Para el caso del modelo booleano, se retorna un conjunto de documentos (sin orden de relevancia) que satisfacen la consulta. Dado que la misma constituye una fórmula lógica bien formada, se considera que el usuario es consciente de los documentos que quiere recuperar, por lo que no se consideró realizar expansión de consulta para este modelo.

4. Evaluación del sistema

Para la evaluación del sistema se ejecutaron ambos modelos del sistema en dos colecciones de datos distintas. La primera colección dentro del fichero cran-1400 y la segunda en 20news-18828. En la primera colección el modelo vectorial obtuvo los valores de medida mostrados por los siguientes gráficos para un total de 255 consultas (Figuras 1, 2 y 3):

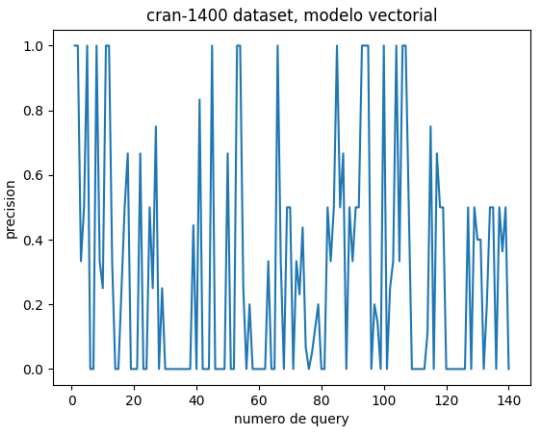


Figura 1: Precisión promedio de 0.3

Aunque el promedio de dichas medidas es bajo, se puede apreciar en los gráficos, que los valores de precisión, recobrado y la medida F1 no tienen un rango bien definido, sino que varían en dependencia de la formulación de la consulta.

Considere la consulta:

```
what are the structural and aeroelastic
problems associated with flight of high
speed aircraft .
```

Al realizar el proceso de búsqueda la mejor solución que encuentra el sistema es la presente en el documento 12.txt que coincide efectivamente con el archivo de mayor relevancia, con un “score” de

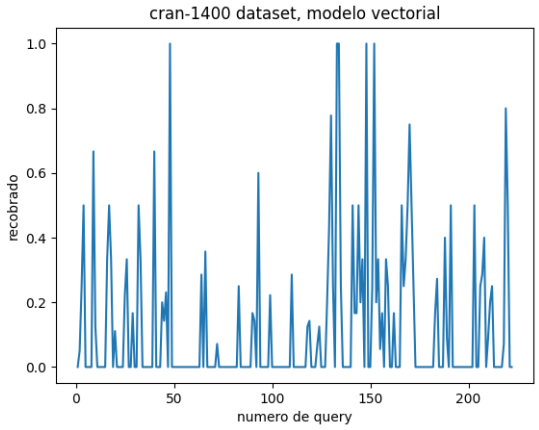


Figura 2: Recobrado promedio de 0.12

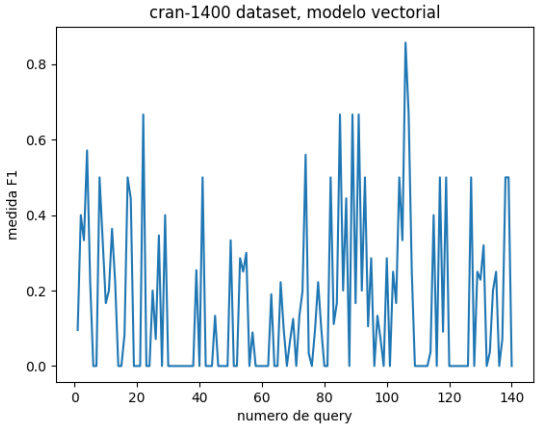


Figura 3: Medida F1 promedio de 0.16

0.51, el resto de los documentos tienen una puntuación inferior al 0.2 por lo que no son considerados de mucha relevancia. Las medidas obtenidas para esta búsqueda fueron de 1.0 de precisión, 0.05 de recobrado y 0.1 como medida F1. Tal como se puede apreciar aunque el archivo de mayor relevancia fue encontrado, solo una pequeña parte de los relevantes fueron recuperados.

En el caso del modelo booleano los valores de sus medidas pueden cambiar drásticamente teniendo en cuenta los operadores lógicos que se utilicen. Por una parte al usar el operador AND pueden ocurrir dos casos, el primero, que no se encuentre la consulta, por lo cual sus medidas para dicha búsqueda descienden y, el segundo que recupere documentos, en este caso, la medida de precisión será bastante alta, pues la mayoría de los archivos recuperados serán relevantes. Por otro lado al usar OR ocurre lo contrario, la medida de precisión disminuye y la de recobrado aumenta.

Los graficos mostrados corresponden a las consultas realizadas usando el operador lógico AND (Figuras 4, 5, y 6):

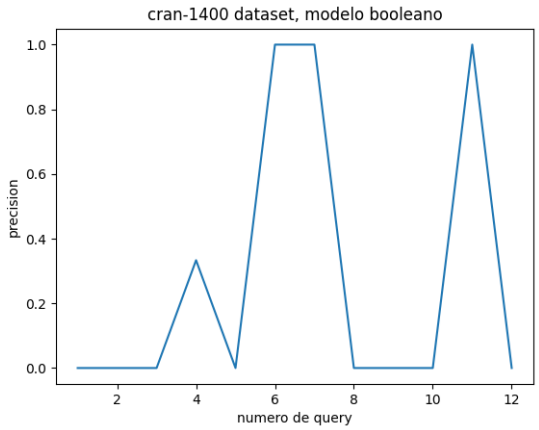


Figura 4: Precisión promedio de 0.27

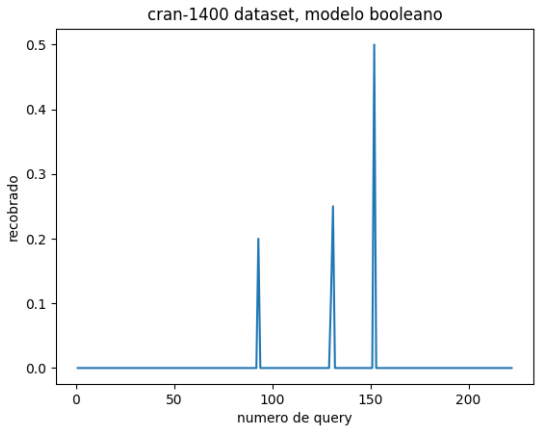


Figura 5: Recobrado promedio de 0.005

Suponga que se desea realizar una consulta en el modelo booleano:

```
it AND method AND solv AND linear AND ellipt AND
diff AND equ AND rapid AND converg
```

Para la consulta anterior, se recupera un solo documento: el archivo 1088.txt, con una puntuación de 1.0, recobrado de 0.5 y por tanto una medida F1 de 0.67.

En la segunda colección de documentos (20news-18828) el modelo vectorial obtuvo los siguientes resultados para diez consultas y un total de 1000 documentos (Figuras 7, 8 y 9):

En este caso las consultas se realizaron sobre una colección de datos previamente clasificada por temas, por lo que los contenidos de los mismos al

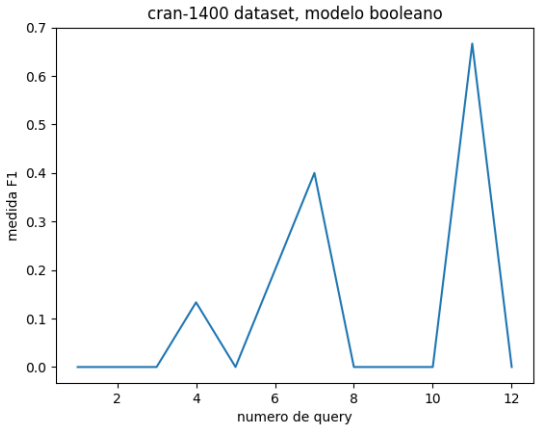


Figura 6: Medida F1 promedio de 0.11

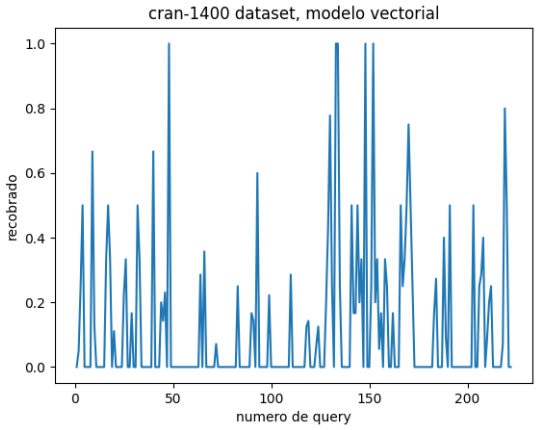


Figura 8: Recobrado promedio de 0.01

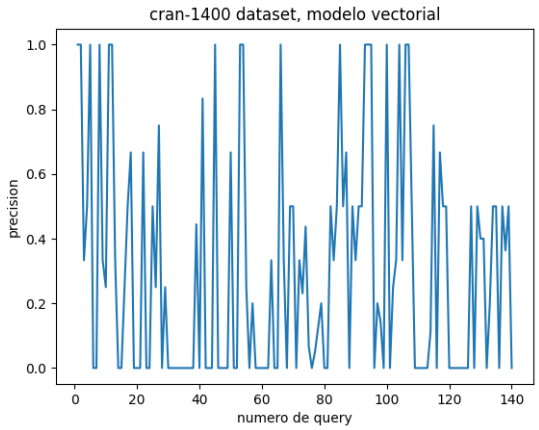


Figura 7: Precisión promedio de 1.0

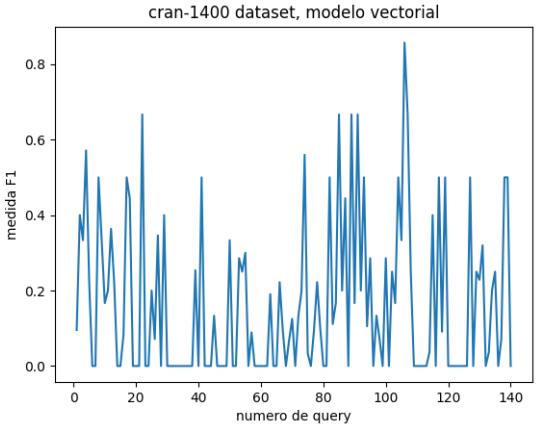


Figura 9: Medida F1 promedio de 0.03

tratar sobre temas distintos, permiten una mayor precisión en el proceso de recuperación, no ocurre así con el recobrado, pues dentro de cada clasificación se encuentra un gran número de documentos relacionados por un mismo tema que el modelo vectorial no es capaz de identificar y por tanto solo recupera una pequeña parte del total de archivos relevantes.

Esto se puede apreciar mejor a través del siguiente ejemplo, dada la consulta:

```
look recommand good royal fre graph libr
pack c c program
```

Se puede apreciar que los documentos relevantes se encuentran en su mayoría dentro del grupo comp.graphics. Para la consulta anterior, el modelo vectorial obtuvo como resultado los documentos 37926 y 37928 como los dos documentos de mayor relevancia, obteniendo en este caso una precisión de 1.0 y como ya se había mencionado anteriormente sobre el recobrado, solo un 0.01 del mismo, para una medida F1 de 0.02.

Por último se analizará nuevamente los resultados que obtiene el modelo booleano para esta segunda colección de documentos (Figuras 10, 11 y 12):

Tal como ocurría con el vectorial, los valores de precisión son elevados mientras que los de recobrado son muy pequeños y teniendo en cuenta que el modelo booleano al emplear el operador AND aumenta esta diferencia, se obtiene como consecuencia los gráficos mostrados.

Un ejemplo se muestra a través de la consulta:

```
icon AND program AND man AND alias
```

El sistema recupera el documento 66430 a partir de dicha consulta, sin embargo los valores de las medidas son de cero, esto se debe a que, aunque encontró un documento con dichos términos, no

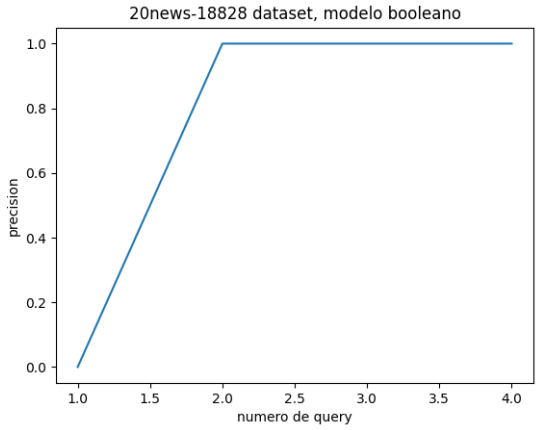


Figura 10: Precisión promedio de 0.75

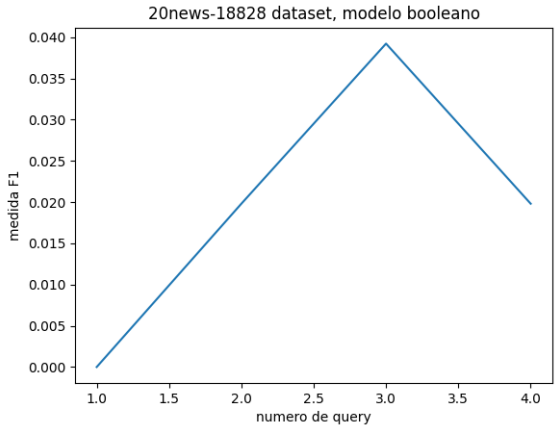


Figura 12: Medida F1 promedio de 0.02

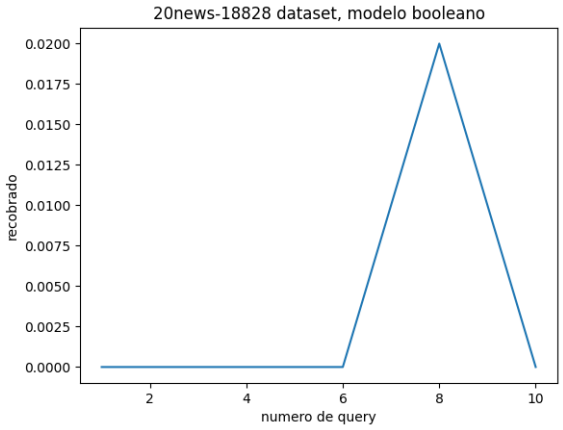


Figura 11: Recobrado promedio de 0.004

constituye un documento relevante (no está dentro del grupo de clasificación de documentos relevantes).

5. Análisis del sistema

A partir de los resultados obtenidos anteriormente es posible apreciar los puntos fuertes y débiles de cada uno de los modelos empleados en el sistema. En el caso del modelo vectorial, el empleo de una variable de suavisado en el cálculo de los pesos de los términos de la consulta, permite obtener resultados más acorde a los elementos de la consulta de ser cercano a 1, mientras que un valor cercano a 0 brinda la posibilidad de encontrar documentos asociados a la consulta, pero que no posea alguno de los términos de esta. El sistema de ranqueo proporcionado por este modelo le proporciona al usuario un orden de relevancia a los documentos recuperados según su consulta, lo cual le facilita la revisión de los mismos para cumplir con su objetivo de búsqueda. Sin embargo, el costo de procesamiento aun es elevado, debido a la gran cantidad de información que se necesita procesar cada vez que se ejecute el sistema la cantidad de memoria usada para ello es enorme, limitando la cantidad de archivos que se puedan tener en la colección. Como se pudo apreciar en los resultados obtenidos durante la evaluación del sistema, es necesario añadir más herramientas al modelo, tales como las opciones avanzadas del lenguaje, que permitan mantener valores más estables de precisión y recobrado.

Por otro lado, el modelo booleano es más facil de construir computacionalmente, aunque la cantidad de datos a procesar y el espacio en memoria es equivalente al modelo vectorial. Permite obtener resultados más consistentes para una consulta con un alto nivel de precisión, sin embargo en muchas ocaciones la cantidad de archivos recuperados es muy grande y se inundaría de información al usuario. Para ello, una posible solución sería el desarrollo de un modelo booleano extendido.

En general, es posible encontrar los documentos relevantes a una consulta en un tiempo de búsqueda razonable a pesar del enorme costo de procesamiento inicial, ya sea por uno u otro modelo. La expansión de consulta es una posible alternativa usada en el sistema para guiar al usuario a formulaciones de consultas más acorde al contenido de los documentos, lo cual facilita de cierta manera la recuperacion de archivos relevantes. Cada uno de los modelos implementados tiene sus condiciones de uso, aun asi depende del usuario decidir en que circunstancias es recomendable el uso de cada uno de ellos y, de hacerlo correctamente, el

sistema terminará constituyendo una gran herramienta para la recuperación de información.

6. Recomendaciones

Existen muchas herramientas con las cuales se podría lograr un mejor desempeño del sistema de recuperación, desde el procesamiento de texto hasta la optimización de la indexación de los documentos. Hoy en día una de las técnicas más usadas es la aplicación de aprendizaje para la expansión de consultas, uno de estos algoritmos es BERT, el cual constituye uno de los principales métodos usados para la inferencia de palabras en oraciones.

En su mayor parte las técnicas de inteligencia artificial y preocesamiento del lenguaje natural juegan un papel crucial en la creación de sistemas de recuperación de información, al incluir el análisis semántico tanto de los documentos como de las consultas.

Otro método con el cual se podría mejorar este sistema es incluyendo operaciones del lenguaje avanzadas, aunque se presentó una alternativa con el análisis de los sinónimos de los términos de la consulta, aún no es un método que se explota del todo.

Referencias

[1] Lic. Alberto Camaraza Monserrate *Recuperación de información: reflexiones epistémicas de una ciencia en su estado embrionario* ACIMED v.13 n.6 Ciudad de La Habana nov.-dic. 2005. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1024-94352005000600010