

Proyecto final de Sistemas de Recuperación de Información

Rodrigo García, Jorge Mederos, Javier Valdés

MATCOM, Universidad de La Habana, Cuba
<http://matcom.uh.cu>

Abstract. El presente trabajo pretende describir el proceso de diseño e implementación de un Sistema de Recuperación de información. El mismo comprende todas las etapas del proceso de recuperación: El procesamiento de la query hecha por un usuario, la representación de los documentos y la consulta, el funcionamiento del motor de búsqueda y finalmente la obtención de resultados. El modelo seleccionado para solucionar el problema es el Modelo Vectorial.

Keywords: Modelo Vectorial, bm25, Sistema de Recuperación de Información, fastText, Recuperación de Información.

1 Introducción

La recuperación de información es un problema existente incluso antes del inmenso crecimiento de la digitalización en la época actual. Se podría afirmar que sus orígenes están en antiguas bibliotecas y centros de documentación en los que se requerían búsquedas bibliográficas de libros o escritos. El objetivo principal de cualquier centro de este tipo es el almacenamiento seguro de la información para satisfacer las necesidades informativas del hombre, de forma veraz, eficiente y pertinente. Esto no ha cambiado con los años, pero las bases de datos, ahora digitales, crecen enormemente en tamaño y se ha hecho necesaria la mejora constante de los sistemas encargados de recuperar la información y satisfacer las necesidades de los usuarios.

Los usuarios traducen su necesidad informativa en una consulta adecuada al sistema. Por tanto deben proveer al mismo de un conjunto de términos que expresen semánticamente aquello que desean obtener. El sistema deberá, dada una base de datos y una consulta, proveer al usuario de los documentos más relevantes para sus necesidades. Cómo calcular dicha relevancia varía en dependencia del modelo seleccionado por los desarrolladores de los Sistemas de Recuperación de información.

En el presente documento se seleccionaron dos modelos:

- El Modelo Vectorial. Ampliamente utilizado en operaciones de recuperación de información, así como en otros campos como la categorización automática o el filtrado de información.
- El Modelo bm25. Un modelo más inteligente y rápido. Una mejora con respecto al tf-idf clásico que, al combinarse con fastText y nmslib, se obtiene un modelo altamente eficiente con grandes resultados.

2 Diseño del sistema

Procesamiento de textos

Antes de tokenizar lo primero que se hace es quitar las contracciones, eso se hace en denoiser, despues se tokeniza usando el tokenizer de nltk, y el la normalizacion hay varios pasos:

- 1- se remueven caracteres que no sean ASCII tipo emojis y demas,
- 2- se pasa todo a lowercase
- 3- se remueven los signos de puntuacion ya que no aportan nada en los modelos
- 4- se llevan los numeros a palabras para normalizar, con esto si buscas “2 cohetes” y “dos cohetes” es lo mismo
- 5- se quitan las stopwords, es decir las palabras que no aportan ningun significado como “the, a, no” etc

Ya lo otro es opcional y seria el stemmer y el lemmatizer Los algoritmos de stemming funcionan cortando el final o el principio de la palabra, teniendo en cuenta una lista de prefijos y sufijos comunes que se pueden encontrar en una palabra flexionada. Este corte indiscriminado puede tener éxito en algunas ocasiones, pero no siempre, por lo que este enfoque presenta algunas limitaciones.

Lemmatizing por otro lado, toma en consideración el análisis morfológico de las palabras. Para hacerlo, es necesario tener diccionarios detallados que el algoritmo pueda consultar para vincular la forma de nuevo a su lexema

Modelo vectorial

Una de las ventajas del modelo vectorial es su utilidad para ponderar términos, permitiéndonos crear un ranking ordenado a partir de la relevancia calculada de cada documento, a partir de su similitud con respecto a la query. Además, facilita el proceso de retroalimentación en el que los usuarios juzgan si los documentos recuperados son o no satisfactorios.

En este modelo se intenta recoger la relación de cada documento D_i , de una colección de N documentos, con el conjunto de las m características de la colección. Formalmente un documento puede considerarse como un vector que expresa la relación del documento con cada una de sus características.

$$D_i \rightarrow \mathbf{d}_i = (c_{i1}, c_{i2}, \dots, c_{im})$$

El vector identifica en qué grado el documento D_i satisface cada una de las m características. Cada c_{ik} sirve para identificar en qué grado el documento D_i posee la característica k . La palabra “característica” puede sugerir un rango bastante amplio de interpretaciones. Podemos concretarlo, por ejemplo, en la ocurrencia de determinadas palabras o términos en el documento.

Es necesario entonces procesar como vectores tanto a los documentos como a las consultas para poder trabajar sobre ellos. Una vez obtenidos los vectores,

se puede calcular una función de similitud que expresará la relevancia de cada documento. Esta función de similitud se explicará más adelante.

Este procesamiento sobre queries y documentos comienza con la tokenización, a partir de la cual se obtienen los términos de los textos y se eliminan caracteres como símbolos de puntuación o espacios. Además, en orden de determinar la similitud de un documento y una consulta, es necesario discernir en qué medida los términos de los documentos son útiles para la consulta en cuestión. No todas las palabras contribuyen con la misma importancia a la caracterización de un documento. Existen palabras casi vacías de contenido semántico, como los artículos, preposiciones y conjunciones. Pero también son poco útiles aquellas palabras que por su frecuencia de aparición en toda la colección de documentos pierden su poder de discriminación. Por tanto, se aplica sobre los tokens obtenidos, técnicas de eliminación de stopwords (palabras sin contenido semántico), stemming (reducción de palabras con significados similares, pero diferencias en la escritura), lematización (reducir palabras de una misma familia en su *lema* correspondiente). El *lema* es la forma que se acepta por convenio como representante de la familia).

Una vez obtenidos los términos caracterizadores de la colección de documentos, comienza la fase de vectorización donde se obtendrá el "peso" de los términos en cada documento.

para determinar la capacidad de representación de un término para un documento dado, se computa el número de veces que este aparece en dicho documento, obteniéndose la frecuencia del término en el documento (*tf* por sus siglas en inglés, term frequency). Por otro lado, si la frecuencia de un término en toda la colección de documentos tiene un valor extremadamente elevado, se opta por eliminarlo del conjunto de términos de la colección (eliminación de stopwords), por lo que podría decirse que la capacidad de recuperación de un término es inversamente proporcional a su frecuencia en la colección de documentos. A esto se le conoce como frecuencia inversa del documento (*idf* por sus siglas en inglés, inverse document frequency).

Se tienen las siguientes fórmulas para calcular el *tf* y el *idf*:

$$tf_{ij} = \frac{freq(i, j)}{max_l * freq_{ij}}$$

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

Luego, el peso w_{ij} de un término t_i en el documento d_j se calcula de la siguiente manera:

$$w_{ij} = tf_{ij} * idf_i$$

También es necesario calcular los pesos w_{iq} de los términos i en las consultas q . Esto se consigue de la siguiente forma:

$$w_{iq} = (a + (1 - a) \frac{freq_{iq}}{max_i freq_{iq}}) * \log \frac{N}{N_i}$$

Finalmente podemos definir la función de similitud mencionada anteriormente y mediante ella, ser capaces de ordenar los documentos según su relevancia. La función de similitud utiliza el coseno del ángulo formado entre los vectores que representan a los documentos y a las consultas.

$$sim(q_i, d_j) = \frac{\sum_{i=1}^n w_{ij} * w_{iq}}{\sum_{i=1}^n w_{ij}^2 \sum_{i=1}^n w_{iq}^2}$$

Un modelo de recuperación de información, de forma general, se compone de un cuádruplo $[D, Q, F, R(q, d)]$ donde:

- D : Es un conjunto de documentos de la colección que conforman el corpus.
- Q : Es un conjunto de consultas que el usuario realiza al sistema.
- F : Es un framework para modelar los documentos de la colección, las consultas y sus relaciones.
- R : Es una función de ranking que asigna a cada par (consulta, documento) un valor acorde a la relevancia del documento para esa consulta.

Luego, el modelo vectorial aquí definido, se define formalmente de la siguiente forma:

- D : Vectores de los pesos asociados a los términos de los documentos.
- Q : Vectores de los términos asociados a los términos de la consulta.
- F : Espacio n-dimensional y operaciones entre vectores del álgebra lineal.
- R :

$$R(q_i, d_j) = sim(q_i, d_j) = \frac{\sum_{i=1}^n w_{ij} * w_{iq}}{\sum_{i=1}^n w_{ij}^2 \sum_{i=1}^n w_{iq}^2}$$

Modelo bm25

Primeramente, se explicará cómo se combina tf-idf con vectores de palabras para poder obtener salidas que son fáciles de interpretar y al mismo tiempo pueden capturar las sutiles relaciones semánticas existentes en el lenguaje. Para esto se utiliza el método denominado fastText.

FastText divide las palabras utilizando caracteres de n-gramas. Por ejemplo, "guitarra" se dividiría en: "gui", "uit", "ita", "tar", "arr", "rra". Para $n = 3$. Esto permite inferir palabras fuera del vocabulario a partir de la superposición de algunos n-gramas. Además, provee cierta robustez con respecto a los errores de ortografía y typos.

Una vez tokenizados los textos, se utiliza la librería Gensim para entrenar el modelo fastText y está todo listo para aplicar tf-idf a los vectores.

FastText es increíblemente efectivo para capturar las relaciones entre palabras

dentro del corpus.

Luego están listas las condiciones para comenzar a hablar de "bm25" (del inglés Best match 25). Este modelo puede ser entendido como un tf-idf con "esteroides", implementando dos mejoras claves:

- Saturación de la frecuencia de términos: bm25 proporciona retornos decrecientes por la cantidad de términos que se comparan con los documentos. Esto es bastante intuitivo, si se quiere buscar un término específico que es muy común en los documentos, entonces se debería llegar un punto en el que el número de apariciones de este término sea menos útil para la búsqueda.
- Longitud de documentos: bm25 considera el tamaño de los documentos en el proceso de comparación. Por ejemplo, si un artículo corto contiene la misma cantidad de términos que matchean que un artículo largo, entonces el corto debe ser más relevante.

Estos cambios además introducen dos hiperparámetros para ajustar el impacto de estos elementos en la función de clasificación. 'k' para ajustar el impacto de la saturación de términos y 'b' para ajustar la longitud del documento.

Finalmente la fórmula quedaría de esta forma:

$$IDF * \frac{(k + 1) * tf}{tf + k * (1 - b + b * dl_{adl})}$$

De esta forma, términos más frecuentes tendrán pesos más altos, pero sólo hasta un valor máximo "k". Y documentos más cortos tendrán pesos mayores siendo dl_{adl} el tamaño del documento actual dividido entre el tamaño promedio de todos los documentos y b un parámetro para ajustar el impacto del tamaño de los documentos.

Entonces, usando el modelo bm25 se quiere lograr crear un mecanismo de búsqueda lo más "inteligente" posible. ¿Pero qué significa "inteligente"? Retornar resultados relevantes a un usuario incluso si este no buscó las palabras específicas resultantes, tener la capacidad de escalar a conjuntos de datos de tamaños mayores, tener una velocidad de búsqueda relativamente alta y ser capaz de manejar de forma inteligente los errores de ortografía y typos. Esto lo logramos con, además de fastText y bm25, la librería nmslib (Non-Metric Space Library) para la rápida búsqueda de resultados.

Una vez obtenido el vector de palabras con fastText, se le aplica entonces bm25 para obtener un vector por cada documento del conjunto de datos. Además, estas técnicas también se pueden utilizar para crear un vector a partir de las queries de los usuarios. Pero, ¿Cómo retornar entonces los resultados relevantes a partir de las queries? Es necesario encontrar los vectores más cercanos al vector de la query y con dimensiones largas, puede resultar ineficiente. Aquí entra nmslib, una solución extremadamente rápida. Utilizando esta librería se puede realizar la búsqueda de una forma mucho más eficiente que con una aproximación por fuerza bruta.

Interfaz Visual

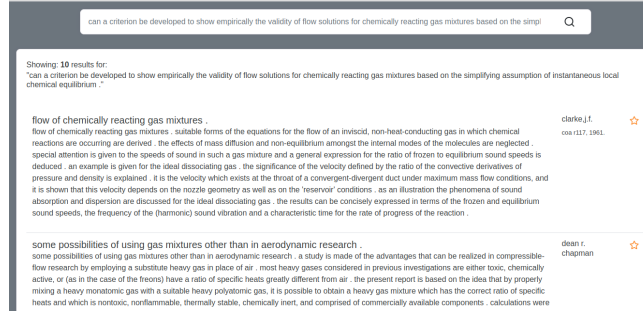


Fig. 1. Interfaz visual

En Fig.1 se muestra una captura de la interfaz visual. Al hacer una búsqueda, se muestran los mejores resultados obtenidos por el modelo. Los símbolos de estrellas a la derecha de cada documento constituyen botones que pueden ser pulsados para que el usuario puntúe un documento recuperado a partir de una query. El sistema de feedback permite actualizar los modelos para acomodarse a los usuarios.

2.1 Detalles de implementación

Para levantar la web, se debe ejecutar el script *main.py* (que se encuentra en la raíz del proyecto) introduciendo dos parámetros. En caso de querer recalcular y reiniciar el modelo vectorial, se introduce 1 como primer argumento, o cualquier otra cosa en caso contrario. Para recalcular y reiniciar el modelo bm25 se introduce 1 como segundo argumento, o cualquier otro valor en caso contrario. Al no recalcular los modelos, se cargan los modelos salvados por última vez. Estos se encuentran en la dirección *models_saves*.

En la carpeta *modules* se encuentra el código. Dentro de esta, las carpetas *data_loader* y *data_processor* contienen las funciones que se utilizan para tokenizar los textos y procesarlos. En *models* están dos archivos con las implementaciones de cada modelo.

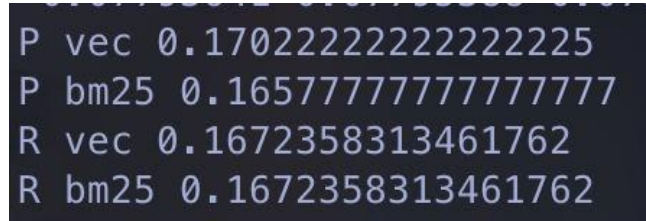
3 Evaluación

Para probar la efectividad de los modelos se utilizó el dataset Cranfield. Las medidas objetivas utilizadas fueron

- Precisión (P)
- Recobrado (R)

- Éxito o porcentaje de consultas en que se recuperó al menos un documento relevante (E)

Los resultados se ven en la Fig. 2



```

P vec 0.17022222222222225
P bm25 0.16577777777777777
R vec 0.1672358313461762
R bm25 0.1672358313461762
  
```

Fig. 2. Interfaz visual

4 Conclusiones

Con este trabajo se ha diseñado e implementado dos modelos para la recuperación de información a partir de consultas sobre un conjunto de datos. Se utilizó un modelo vectorial sencillo y el bm25 como una mejora. A partir de la evaluación de los modelos se pudo ver el éxito de estos y realizar una comparación entre los mismos. Además, se implementó una interfaz visual para la introducción de nuevas consultas a partir de la cual se obtienen los resultados ordenados según el ranking devuelto por los modelos.

References

1. Sistemas de Recuperación de Información. Departamento de Programación, Facultad de Matemática y Computación, Universidad de la Habana 2022. Prof. Carlos Fleitas Aparicio, Prof. Marcel E. Sánchez Aguilar.
2. <https://towardsdatascience.com/>
3. <https://www.nltk.org/>
4. Wikipedia
5. <https://medium.com/>