Universidad de La Habana

Facultad de Matemática y Computación

Sistemas de Recuperación de Información

Proyecto Final

Autores:
Olivia González Peña C511
Juan Carlos Casteleiro Wong C511

Resumen

Este proyecto se centra en el diseño, implementación, evaluación y análisis de un Sistema de Recuperación de Información. El sistema comprende todas las etapas del proceso de recuperación de información, desde el procesamiento de la consulta realizada por el usuario, la representación de los documentos y la consulta, el funcionamiento del motor de búsqueda y la obtención de los resultados.

El sistema contiene implementados distintos modelos de recuperación de información, como el vectorial y el booleano clásicos e incluye otros que se detallan más adelante.

La interacción con el sistema se realiza a través de una interfaz visual sencilla que facilita el entendimiento de las funcionalidades que oferece el sistema al usuario.

Palabras Clave

sistema de recuperación de información, motor de búsqueda, recuperación de documentos

Detalles

Para el análisis de las consultas y los documentos, es necesario llevar a cabo un preprocesamiento de los datos, en el cual, primeramente, se realiza la tokenización de las cadenas de texto, separando en tokens los posibles términos indexados.

Como segundo paso se pueden realizar el proceso de lematización o el de stemmig. El lema, por convenio, es la forma que se acepta como representante de las distintas formas flexionadas en las que puede aparecer una palabra o término [4]. El stemming es un método para reducir una palabra a su raíz o (en inglés) a un stem [5]. El sistema contiene una implementación para cada uno de estos procesos, debe seleccionarse con el que se desea trabajar. Por defecto tiene integrado la lematización como parte del preprocesamiento de los datos.

El resultado es sometido al siguiente paso del preprocesamiento, que es la eliminación de las "stopwords", que no son más que aquellas palabras que por sí solas carecen de significado semántico (como artículos, conjunciones y preposiciones) y, por tanto, no son interesantes como términos indexados. Finalmente, se culmina eliminando los signos de puntuación.

Los N términos resultantes luego del preprocesamiento, conformarán el conjunto de términos indexados. En consecuencia, los documentos se representan como vectores de tamaño N, que en la i-ésima posición tienen el número correspondiente al peso de dicho término en el documento. Igualmente, las consultas son representadas a través de vectores con esta forma.

El sistema contempla que pueda ser utlizado por usuarios expertos y no expertos, permitiendo que se seleccione el modelo booleano clásico, más adecuado para los primeros, y el vectorial clásico que está más orientado a los del segundo tipo. Estos, y otros modelos como el LEM y el TF-IDF se incluyen para las dos colecciones dadas (Cranfield y MED) con el objetivo de que se puedan comparar los resultados obtenidos para el mismo corpus con diferentes modelos.

Modelo Vectorial Clásico

El modelo vectorial clásico se encuentra en el fichero *vector_space_model.py*, el cual a través del método *compute_ranking* (que recibe una consulta) devuelve una cantidad definida anteriormente (*RANKING_COUNT*) de documentos ordenados por relevancia a la consulta. Para ello se calcula la similitud entre el

vector que representa a la consulta y los vectores de cada documento, computando el peso de los términos de la consulta y el peso de los términos en los documentos. Para calcular el valor de cada componente de los vectores de las consultas se agrega una constante de suavizado (*ALPHA*).

En el fichero *inverted_index.py* se provee la clase *InvertedIndex*, que una vez dada la colección computa la frecuencia de cada término en cada documento, la cantidad de documentos en los que aparece un término, la mayor frecuencia de aparición de un término en un documento, etc., datos necesarios para computar el tf e idf en el modelo vectorial.

En el fichero *cranfield_parser.py* se encuentra la implementación de la clase *CranfieldParser*, la cual tiene como objetivo estructurar los documentos de la colección Cranfield para su posterior análisis. En el fichero *dataset*, se encuentra la implementación de la clase *Dataset* para representar los documentos de la colección.

La clase *VectorSpaceModel* aúna las funcionalidades expuestas anteriormente, su constructor recibe un objeto de tipo *Collection* que representa los documentos del corpus.

El método *get_ranking_index* recibe la consulta y el número por el que se desea truncar el ranking y devuelve una lista de enteros que son los índices en la colección de los documentos considerados relevantes.

Los métodos *generate_query_vector* y *generate_document_vector* se encargan de crear el vector de la consulta a partir del string de esta, y el vector del documento a partir del índice del mismo, respectivamente.

La clase *Evaluator* permite evaluar el comportamiento de los modelos empleando tres métricas: Precisión, Recobrado y Medida F1, cuyos valores se obtienen con los métodos *compute_precission*, *compute_recall* y *compute_f1*, respectivamente.

Los resultados de las métricas se encuentran condicionados por dos factores. El primero es la cota inferior que se establece para la similitud entre el documento y la consulta, y el otro es la cantidad máxima de documentos que se permite devolver de los que cumplen con el primer requisito.

Modelo Booleano

Se incluye la implementación de un modelo de recuperación de información booleano clásico en el archivo *models.py*. Este, a diferencia del resto de los modelos que contiene el sistema, trabaja con vectores de pesos booleanos, indicando simplemente la existencia o no de los términos en los documentos y las consultas.

Este modelo también requiere el preprocesamiento de los datos de los documentos y las consultas mencionado anteriormente.

Para analizar las consultas, son separadas las componentes conjuntivas que la conforman, tomando como separador el símbolo para las disyunciones, que en este caso es "or" dado que las colecciones con las que se trabaja están escritas en inglés. Las componentes son analizadas individualmente para analizar la similitud con los documentos, de forma que resultan relevantes a la consulta solo los que tengan correspondencia total de los términos. En consecuencia, tampoco existe un concepto de ranking para los resultados del sistema cuando se utiliza este modelo.

Otros modelos

El sistema incluye otros dos modelos de recuperación de información: TF-IDF y LEM. La implementación de cada uno de ellos se recoge en una clase, que heredan, a su vez, de la clase *IRSystem* que se inicializa con el corpus de documentos y el identificador del modelo en cuestión, es decir, 2 indicará que es el modelo TF-IDF y 3 para LEM. Para la realización del procesamiento de los documentos y las consultas se apoya en la biblioteca Gensim [3] y las funcionalidades que la misma contiene en los módulos *models.tfidfmodel* [2] y *models.logentropy_model* [1].

Similar a como ocurre con el modelo vectorial explicado anteriormente, los datos son preprocesados como primer paso del proceso de recuperación. De igual forma, los métodos *create_document_vector* y *create_query_vector* se utilizan para preparar los términos de los documentos y las consultas, respectivamente.

En el método *ranking_function* se calcula, mediante el módulo *similarities* de Gensim, la matriz de similitud asociada a los documentos.

Resultados

En la aplicación mediante la cual se interactúa con el sistema, se incluye un apartado para la visualización de los resultados de evaluar todos los modelos implementados con las dos colecciones de documentos con las que se trabaja, permitiendo que se puedan comparar los mismos.

Para agilizar la obtención de los resultados, una vez analizados los datos de las colecciones, asumiendo que no sufrirán variaciones, son almacenados en archivos cuyos nombres indican la procedencia del contenido, en la carperta *data*. De esta forma, la primera vez que sean procesados los datos el sistema tardará un poco más en obtener los resultados para mostrar al usuario, y a partir de entonces, se reutilizarán los cómputos realizados para futuras operaciones.

Recomendaciones

Como futuras mejoras sería útil agregar al sistema funcionalidades como la expansión de consulta o la retroalimentación.

Referencias

[1] Radim Rehurek. models.logentropy_model - logentropy model, 2022.

[2] Radim Rehurek. models.tfidfmodel - tf-idf model, 2022.

[3] Radim Rehurek. Topic modelling for humans, 2022.

[4] Wikipedia. Lematización, 2022.

[5] Wikipedia. Stemming, 2022.