Universidad de la Habana, MATCOM
Sistemas de Recuperación de Información (2022)

Informe de Entrega Parcial

Samuel David Suárez Rodríguez C512
Enmanuel Verdesia Suárez C511
Gabriel Fernando Martín Fernández C511

1. Descripción General

1.1. Resumen

Se pretende desarrollar un Sistema de Recuperación de Información (SRI) con el apoyo de técnicas de Inteligencia Artificial (IA). Dado un conjunto de varios documentos de diversos temas se desarrollará una aplicación de búsqueda que obtenga resultados relevantes y acordes a una consulta presentada por el usuario. Además se emplearán métodos para clasificar los documentos y extraer sus características más relevantes usando algoritmos para la clasificación y selección de características. Todos estos procesos de clasificación se apoyarán en el uso de algoritmos de machine learning (ML) para lograr su objetivo.

1.2. Estructura del proyecto

El proyecto presenta tres componentes principales: el backend o componente lógica de la aplicación, junto con la API, y el frontend en forma de un servicio web.

Está desarrollado en GitHub bajo la organización Gologle. Ahí pueden ser encontrados los dos repositorios principales. Además están hospedados el cliente del sistema y la API en internet.

- Web: https://gologle.vercel.app
- API: https://gologle-api.herokuapp.com/

Detalles de implementación

2. Procesamiento de los sets de datos

Para cada set de datos empleado fue implementado un parser para extraer su información. Este proceso no se pudo realizar de forma común pues cada set contenía los documentos de forma particular. Una vez *parseados* los sets de datos se puede delimitar e identificar cada documento que contiene de forma única por un ID y así guardar el texto de los documentos en una base de datos SQLite. Dicha base de datos facilita obtener un menor tiempo de respuesta ante un pedido de parte del cliente.

Los sets de datos con los que se trabajó fueron Cranfield (1400 documentos), Newsgroups (18828 documentos) y Reuters (????? documentos). Se contaba con consultas y resultados relevantes para Cranfield. Que se tiene en Newsgroups y/o Reuters?

3. Modelo Vectorial (TF-IDF)

Este modelo esta basado en la representación de los documentos mediante *bag-of-words*. Es decir, se transforma cada documento a una vector de tamaño fijo que solo contiene información de la cantidad de veces que aparece cada palabra por componente. Este modelo a pesar de ser efectivo tiene como debilidad que se pierde la información respecto al orden de las palabras, además, como no aprende el significado de las palabras, la distancia entre vectores no siempre representa una diferencia en el significado.

3.1. Preprocesamiento

Dado que ha sido implementado el modelo vectorial basado en TF-IDF (del inglés, *term frequency* e *inverse document frequency*), haciendo uso de la biblioteca `sklearn` nos ajustamos a la forma que esta requiere para representar los documentos. Dicha biblioteca se usó para *tokenizar* los documentos y extraer como *features* los términos que estos contienen. El texto de los documentos es llevado a minúscula, además los términos de un simple carácter son eliminados. De igual forma fueron eliminadas las *stopwords* del idioma

inglés, conjunto de palabras comunes que no son de relevancia, lo cual permite al modelo enfocarse en palabras más relevantes. Finalmente se realizo lematización sobre el texto usando la biblioteca `nltk`. Así se obtiene cada documento como un vector de términos (palabras) con una dimensión determinada.

3.2. Construcción del índice

A partir del conjunto de documentos se obtiene una matriz esparcida que representa la cantidad de veces que aparece el término i en el documento j. Sobre esta matriz nos apoyamos para calcular la frecuencia de documentos inversa (idf) para cada término.

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

n_i: cantidad de documentos en los que aparece el término i.

Luego haciendo uso de la clase `TfidfTransformer` que se encuentra en el módulo `sklearn.feature_extraction.text` calculamos los pesos para cada término en cada documento, de acuerdo a la expresión:

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

donde t_{ij} es la frecuencia normalizada del término i en el document j,

$$tf_{ij} = \frac{freq_{ij}}{max_l freq_{lj}}$$

Hecho esto, se tiene la matriz esparcida de los pesos, la cual es guardada en la base de datos para agilizar el tiempo de respuesta.

La base de datos creada a partir de los documentos, los términos, la idf para cada término y los pesos es considerada el índice del sistema para satisfacer las consultas de la forma más rápida posible.

3.3. Recuperación de Información

Cuando es recibida una consulta q, esta es procesada de forma similar a como se procesa un documento. Se calcula el peso para cada término (w_{iq}), mediante la siguiente expresión

$$w_{iq} = (\alpha + (1 - \alpha)\frac{freq_{lq}}{max_l freq_{lq}}) * idf_i$$

donde α es un valor de suavizado que permite amortiguar la contribución de la frecuencia de los términos, se le asignó un valor de 0.5.

La similitud entre el vector obtenido y los documentos de la base de datos es calculada usando como valor de referencia el coseno del ángulo entre estos. Los documentos con mayor similitud son dados como resultado de forma ordenada.

4. Modelo Vectorial (Doc2Vec)

Para desarrollar el modelo nos apoyamos en la biblioteca `gensim`, que ofrece una implementación del modelo `Doc2Vec`, que se encuentra en el módulo `gensim.models.doc2vec`. Esta esta basada en *Paragraph Vector* [**?**], un algoritmo no supervisado que es capaz de aprender *features* de tamaño fijo de textos de tamaño variable, como oraciones, párrafos y documentos.

4.1. Preprocesamiento

De igual forma al modelo basado en TF-IDF, aquí se llevo el texto de los documentos a minúscula, fueron removidas las *stopwords*, palabras de una sola letra y se realizó la lematización. El texto obtenido para cada documento fue empleado para crear una instancia de `TaggedDocument`, del módulo `gensim.models.doc2vec`.

4.2. Entrenamiento

Con el conjunto de `TaggedDocument`, cada uno con el ID del documento relacionado, se realizó el entrenamiento del modelo `Doc2Vec`. Este se realizo con vectores de tamaño 50, por 200 epochs. Se probaron vectores de mayor tamaño y número de epochs mayor pero no mejoraron los resultados prácticos.

EL modelo entrenado es persistido para en futuros inicios del sistema solo tener que cargarlo.

4.3. Construcción del índice

Dado que este modelo realiza la inferencia a través del modelo, solo es necesario guardar en el índice los documentos que van a ser devueltos al usuario.

4.4. Feedback

En la base da datos (índice) también fue creada una tabla para registrar el feedback de los usuarios. Para cada documento obtenido de una consulta se puede dar una valoración de si fue relevante o no (1 o -1). Esta valoración es almacenada en el índice para en futuras consultas similares usarla y mejorar la cantidad de documentos relevantes devueltos.

Para calcular el feedback fue empleado el algoritmo de Rocchio [**?**], el cual tiene como premisa hallar el centroide de los puntos relevantes y tratar de alejarse de los no relevantes.

$$q_m = \alpha q_0 + \frac{\beta}{|D_r|} \sum_{d_j \in D_r} d_j - \frac{\gamma}{|D_{nr}|} \sum_{d_j \in D_{nr}} d_j$$

donde:

D_r: conjunto conocido de documentos relevantes

D_{nr}: conjunto conocido de documentos no relevantes

α, β, γ: pesos para consulta, documentos relevantes y documentos no relevantes respectivamente

Se tomaron como valores de referencia $\alpha = 0{,}97$, $\beta = 0{,}4$ y $\gamma = 0{,}15$.

4.5. Recuperación de Información

Cuando se recibe una query antes de hallar la similitud con los documentos del modelo se le aplica Rocchio de acuerdo a lo descrito anteriormente. El vector resultante es el empleado para calcular la similitud con los documentos del modelo. Esta es calculada usando el coseno del ángulo entre los vectores [**?**], de forma similar al primer modelo.

Los resultados son devueltos en un ranking, ordenados por relevancia, nuevas actualizaciones por feedback son procesadas y añadidas a la base de datos.

5. Evaluación de los modelos

5.1. Métricas objetivas

Dado que se tenía las consultas y sus relevancias para el set de Cranfield, se condujo una evaluación de los modelos con la métrica F. Los resultados no fueron buenos al promediar los resultados para las aproximadamente 150 consultas pues muchas no estaban devolviendo resultados relevantes.

Motivo de esto es que algunas consultas tenían pocos resultados relevantes en el set de prueba. En el caso del modelo basado en TF-IDF también las consultas al tener tantos términos el tiempo de respuesta se hacía grande. Doc2Vec, mostró un tiempo de respuesta rápido pero lo antes mencionado sobre la poca cantidad de relevantes en el set de prueba, sumado a la alta especificidad del dataset que causa que documentos relevantes y no relevantes no sean ubicados cerca en el espacio no ofrece resultados satisfactorios.

Para consultas más simples Doc2Vec presenta mejores resultados que el vectorial al detectar la semántica latente debido a su representación.

Para algunas consultas de Cranfield se obtienen los siguientes resultados:

5.2. Métricas subjetivas

Métricas objetivas y subjetivas estudiadas empleando 2 colecciones

Mostrar consultas con los resultados (al menos 1 por colección)

6. Ventajas y Desventajas del Sistema

Aquí creo se puede hablar de todo el sistema en general con un análisis crítico

Recomendaciones

Existen varios acercamientos a los SRI que podrían ser de utilidad para mejorar la calidad de los resultados obtenidos por los

modelos propuestos y su rendimiento. Una posibilidad es agregar un sistema de recomendaciones basado en proponer documentos con características similares a los de los resultados de búsqueda. En este sentido se pueden recomendar aquellos documentos más relevantes que sean miembros de los clústeres a los que pertenecen los resultados. Estas ideas se basan en emplear algoritmos de clasificación y agrupamiento para proponer posibles resultados que escapen a los obtenidos por los modelos implementados. Además para mejorar la velocidad de búsqueda es posible añadir filtros basados en las características de los documentos para así restringir el campo de búsqueda. Esta idea puede mejorar de manera considerable el rendimiento a cambio de cierta pérdida de precisión que dependerá de la calidad de la clasificación obtenida y de la elección de filtro del usuario.

Referencias

1. Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, pp. 1188–1196, PMLR, 2014.
2. "Rocchio algorithm." `https://en.wikipedia.org/wiki/Rocchio_algorithm`, June 2022.
3. "Method most$_s$*imilar*.", June 2022.