Informe Moogle!

Reynol Gomez Franco C121

July 24, 2023

1 Clase BaseDeDatos

La clase `BaseDeDatos` contiene varios campos y métodos para cargar, procesar y almacenar información de documentos. A continuación, se describen los campos y métodos de la clase:

- `separadores`: un arreglo de cadenas que contiene los caracteres que se utilizarán para separar las palabras en los documentos.
- `TFIDF`: un diccionario de diccionarios que contiene los valores de TF-IDF para cada palabra en cada documento.
- `tf`: un diccionario de diccionarios que contiene los valores de TF para cada palabra en cada documento.
- `idf`: un diccionario que contiene los valores de IDF para cada palabra en todos los documentos.
- `words`: una lista de todas las palabras únicas en todos los documentos.
- `VectorsDocument`: un diccionario que contiene los vectores de documentos para cada documento.

2 Metodo TFIDF()

La clase tiene un constructor que inicializa los campos `TFIDF`, `tf`, `idf`, `words` y `VectorsDocument`. El método `Load()` carga los documentos de un directorio y calcula los valores de TF e IDF para cada palabra en cada documento. El método `TF()` devuelve el diccionario de TF para todos los documentos. El método `IDF()` devuelve el diccionario de IDF para todas las palabras en todos los documentos. El método `VectorDocument()` calcula los vectores de documentos para cada documento.

Para calcular el TF-IDF primero se carga en una lista el nombre de todos los documentos con el método `GetFiles()` (tiene predeterminada mi ruta). Luego, se itera por dicha lista convirtiendo cada texto en un string a través de `File.ReadAllText()` y se realiza un segundo bucle para iterar por las palabras

de cada texto usando el método `.Split(separadores)`. A partir de aquí, se quiere sacar la mayor cantidad de información posible, por lo que se crea un diccionario `wordfordocument` con el objetivo de saber en cuántos documentos está una palabra, lo cual nos servirá más adelante para calcular el IDF. Asimismo, se crea un diccionario de frecuencias para saber cuántas veces aparece una palabra en un documento y luego se añade al diccionario `TF` una llave documento y la frecuencia de cada palabra en ese documento. Ahora tenemos un diccionario `wordfordocument` que contiene en cuántos documentos está una palabra, y un diccionario `TF` con la frecuencia de cada palabra en cada documento. A partir de esto, calculamos el IDF con el logaritmo en base 10 de la división de la cantidad total de documentos sobre la cantidad de documentos en los que se encuentra una palabra. Además, el TF se calcula dividiendo la frecuencia de la palabra en el documento sobre la frecuencia de la palabra con mayor frecuencia. Por problemas de optimización y para tratar de hacer la mayor cantidad de cosas en las menos iteraciones posibles, tenemos dos diccionarios en el mismo método, y solo puedo devolver un tipo, por lo que decidí devolver un diccionario que almacene los dos diccionarios para luego separarlos en el constructor.

3 Método Vectores()

Para crear los vectores de cada documento, se itera por cada documento y por la lista de palabras totales de todos los documentos. Si el documento contiene la palabra, se coloca el valor TF-IDF en esa posición, si no la contiene, se coloca 0.

4 Clase Query

La clase `Query` tiene dos campos públicos: `words` y `TFquery`. El constructor de la clase toma una cadena de consulta como entrada y la convierte en una lista de palabras en minúsculas utilizando el método `Split` de la clase `string`. El constructor también llama al método text

ttTF() para calcular la frecuencia de término (TF) de cada palabra en la consulta y almacenarla en el campo `TFquery`.

El método `TF()` es un método privado que toma la lista de palabras de la consulta y calcula la frecuencia de cada palabra en la lista. El método utiliza un diccionario `frecuencias` para almacenar la frecuencia de cada palabra y luego calcula la frecuencia de término (TF) de cada palabra dividiendo su frecuencia por la frecuencia máxima de cualquier palabra en la lista. El método devuelve un diccionario `tf` que contiene la frecuencia de término de cada palabra en la lista.

5 Clase ModeloVectorial

La clase `ModeloVectorial` representa un modelo vectorial utilizado en la recuperación de información. Esta clase tiene varios campos y métodos que se utilizan para calcular la similitud entre un conjunto de documentos y una consulta.

item `documents`: un objeto de la clase `BaseDeDatos` que representa una colección de documentos. item `query`: un objeto de la clase `Query` que representa una consulta. item `VectorQuery`: un arreglo de tipo `double` que representa el vector de consulta. item `Score`: un diccionario que almacena los puntajes de similitud entre los documentos y la consulta. item `MoreFrequency`: una lista que almacena las palabras más frecuentes en los documentos. enditemize

El constructor de la clase `ModeloVectorial` toma dos parámetros: un objeto de la clase `BaseDeDatos` y un objeto de la clase `Query`. Este constructor inicializa los campos `documents`, `query`, `VectorQuery`, `Score` y `MoreFrequency` utilizando los métodos `VectorsQuery()`, `Scores()` y `MoreFrequencyDocuments()`.

El método `VectorsQuery()` calcula el vector de consulta utilizando la frecuencia de términos de la consulta previamente calculada en la clase `Query` y el valor IDF (Inverse Document Frequency) de cada término en la colección de documentos.

El método `Scores()` calcula los puntajes de similitud entre los documentos y la consulta utilizando el método `Similitud()`.

Los métodos `Normalizar()`, `Multiplicar()` y `Similitud()` son métodos auxiliares que se utilizan para calcular la similitud entre dos vectores.

6 Método MoreFrequencyDocuments()

El método `MoreFrequencyDocuments()` devuelve una lista de las palabras más frecuentes en los documentos. Este método ordena el diccionario `frecuencias` por valor de forma descendente, toma las cinco primeras palabras del diccionario ordenado y crea una lista con las palabras más frecuentes.

7 Método VectorQuery()

El objetivo de crear el `VectorQuery` es calcular la similitud de este con los vectores documentos y así devolver los cinco documentos que más se asemejen a la consulta. Para crear este vector, iteramos por la lista de palabras `words` y si la consulta contiene la palabra, le añadimos su valor TF-IDF en esa posición; de otra manera, se inserta un 0.

8 Método Similitud()

Para crear el diccionario `Score` que almacena el nombre del documento y su nivel de importancia con respecto a la consulta, iteramos por el diccionario `VectorsDocument` y hallamos la similitud de cosenos de cada vector documento con el vector de la consulta. Luego, extraemos una lista `MoreFrequencyDocuments` con los cinco primeros documentos del diccionario.

9 Clase Moogle

En la clase `Moogle`, se crea una instancia de la clase `ModeloVectorial` y se le pasa como parámetro las dos instancias de `BaseDeDatos` y de `Query`, y se devuelve la lista con los cinco primeros documentos.