Documentación de Moogle

Francisco Prestamo

2023

1. Introducción

En el proyecto se debe agregar documentos al content antes de iniciar el proyecto, de ser posible que los documentos estén en ingles. Esto es muy importante puesto que así mejora mucho la precisión a la hora de buscar, debido al uso de un diccionario con una gran cantidad de palabras en inglés lematizadas llamado SteammedWords.json el cual no puede ser borrado.

2. Componentes del Proyecto

Mi proyecto cuenta con 4 lugares principales: MoogleEngine, MoogleServer, MoogleLibrary y Algebra Library. En Moogle Library se encuentran las clases creadas por mi para realizar toda la parte de la lógica de obtener documentos procesarlos y todo lo demás.

2.1. Moogle Library

2.1.1. Clase Obteiner

La clase Obteiner va a ser la clase que se llama dentro del Moogle server para llevar a cabo todo desde procesar los textos mediante el uso de la clase Process hasta calcular el tf-idf de los mismos.

2.1.2. Clase Process

Para obtener los textos el Obteiner llama a la clase Process y esta se encarga de primeramente agregar espacios en blanco al final de cada texto y tenerlos de forma organizada en otra carpeta ContentwithSpaces para una mayor organizacion. Ademas se encarga de procesarlos de manera que estos no contengan ningun caracter extraño y queden en minusculas. Tambien se encarga de contar las ocurrencias de cada palabra en cada texto.

2.1.3. Agregar, quitar y editar documentos TXT en tiempo real

Una de las características más útiles de un sistema de búsqueda es la capacidad de agregar, quitar y editar documentos en tiempo real, sin tener que detener o reiniciar el sistema. En el proyecto Moogle, esto se logra mediante el monitoreo constante de los archivos TXT en la carpeta de documentos. Cuando se inicia el proyecto, se lee un archivo JSON que contiene la información sobre los documentos TXT y su ubicación en el sistema de archivos. Si se agregan, quitan o editan documentos en la carpeta de documentos mientras el proyecto está en ejecución, se detectan estos cambios mediante el monitoreo constante de la carpeta. Si se detecta un cambio en los documentos, se vuelven a cargar los documentos TXT y se actualiza la matriz de tf-idf y el índice invertido. Si no se detecta ningún cambio, el proyecto continúa utilizando los datos obtenidos de los archivos JSON. Para agregar un nuevo documento, simplemente se debe colocar el archivo TXT en la carpeta de documentos. El proyecto detectará automáticamente el nuevo archivo y lo agregará a la matriz de tf-idf

2.1.4. Serialización y Deserialización

Para guardar los datos se utiliza un serializador de json, y para obtenerlos luego se utiliza un deserializador.

2.1.5. Fórmula del tf-idf

Se utiliza la siguiente formula de tf-idf:

$$tfidf = \frac{xi}{ni} \times ln\left(\frac{d+1}{t}\right)$$

donde:

- xi representa la cantidad de apariciones de una palabra en un documento,
- ni representa la cantidad de palabras de dicho documento,
- d es la cantidad de documentos,
- t es la cantidad de documentos en que aparece la palabra.

2.2. Procesamiento de Query

Dentro de la clase CosineSimilarityCalculator se realiza todo lo relacionado a obtener todos los operadores presentes dentro del query y según que tipo de operadores es se agrega a una estructura para procesarlos de mejor manera.

2.2.1. Operadores

1. Si es de tipo * se guarda cuantos * hay antes de la palabra por cada palabra con *

2. Si es de tipo (operador mayor que hacia arriba) se guarda en una lista de yesWords para comprobar que la palabra aparezca en la query

3. Si es de tipo ! Se agrega a una lista de noWords pero ademas se retira de la query para evitar las apariciones de palabras semejantes

4. Si es de tipo (cercanía) se agrega a un diccionario las palabras a sus lado para verificar que estén cerca

2.2.2. Procesamiento de palabras con distancia de Levenshtein

En el proyecto Moogle, se utiliza la distancia de Levenshtein para procesar las palabras de la query y encontrar las palabras más cercanas dentro del universo de palabras. Cuando se recibe una query, se separa en palabras y se calcula la distancia de Levenshtein entre cada palabra y todas las palabras del universo de palabras. Se guarda una lista con las palabras más cercanas para cada palabra de la query.

2.2.3. Cálculo de Semejanza

Luego se calcula la semejanza entre el tfidf de la query con el de cada documento mediante el cosine similarity al igual que con los tf-idf de palabras lematizadas pero estos últimos añaden al score solo la mitad.

2.3. Snippet Calculator

Además, con el Snippet calculator busco las apariciones de las palabras que ya son las más parecidas a las de mi universo de palabras y las busco dentro de los textos cuyo score sea diferente de 0.

2.4. La librería AlgebraLibrary y sus clases Matrix y Vector

La clase Matrix representa una matriz de números, y contiene operaciones para multiplicarse con otras matrices, sumarse, restarse, etc. La clase Vector representa un vector de números, y también contiene operaciones para multiplicarse con matrices, sumarse, restarse, etc.

3. Funcionalidad del Proyecto

En el proyecto puedes buscar palabras y te da los documentos que contengan esas palabras y además documentos que contengan palabras parecidas. Pero le da prioridad a los documentos que contengan a las palabras exactas.