Moogle!

1er Proyecto de Programación

Ronald Provance Valladares C121

Facultad de Matemática y Computación
Universidad de la Habana

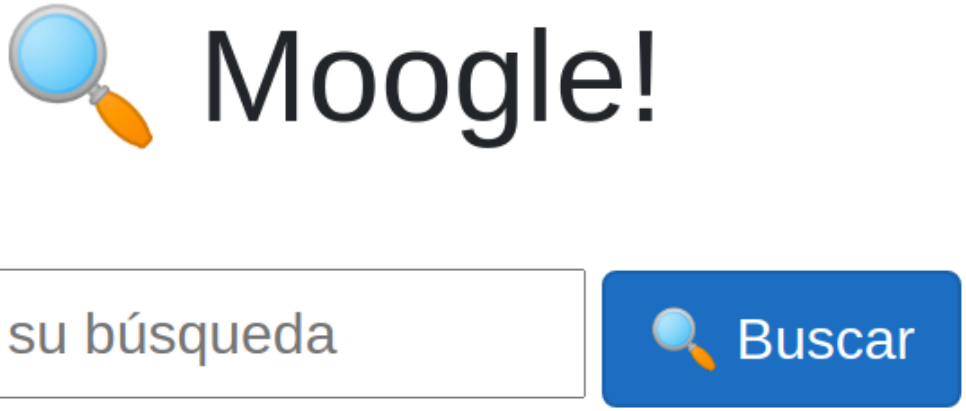
25,Julio,2023

MATCOM

Índice

MATCOM

¿Qué es Moogle?



Moogle! es una aplicación web, cuyo propósito es buscar inteligentemente un texto en un conjunto de documentos, desarrollada con tecnología .NET Core 7.0, específicamente usando Blazor como *framework* web para la interfaz gráfica, y en el lenguaje C#.

MATCOM

Descripción

La aplicación está dividida en dos componentes fundamentales:

- ‘MoogleServer‘ es un servidor web que renderiza la interfaz gráfica y sirve los resultados.
- ‘MoogleEngine‘ es una biblioteca de clases donde está implementada la lógica del algoritmo de búsqueda.

MATCOM

Para llevar a cabo la tarea que se me ha asignado he utilizado lo que se conoce como modelo vectorial, que no es más que un modelo algebraico utilizado para el filtrado, indexado, recuperación y cálculo de relevancia de la información y además, mediante este modelo se pueden explotar las relaciones geométricas entre dos vectores documento (y términos) a fin de expresar las similitudes y diferencias entre términos.

MATCOM

Base de Datos

Toda información solicitada será extraída de la colección de documentos (con extensión .txt) que se ingrese en la carpeta 'Content'.

Interacción con el usuario

- El motor recibe una entrada por parte del usuario, llamado 'query', el cual puede contener algunos operadores que llegarían a optimizar la búsqueda, estos operadores son:

  - * este operador el usuario lo puede ingresar tantas veces como quiera antes de una palabra sin poner espacio, esto le dará mayor importancia o relevancia a los documentos que contengan esta palabra.
  - ! el usuario debe colocarlo junto a la palabra, este negará (no devolverá) aquellos documentos que contengan esa palabra.

Estructura

El proyecto esta divido en diferentes clases que lo componen,de las cuales hablaremos un poco a continuación

Clase Moogle:

En esta clase se encuentran los llamados a las otras clases y métodos antes mencionados para hacer funcionar el buscador, además de un algoritmo de ordenación de los títulos de los documentos y el score para que sean devueltos en orden de mayor a menor score.

Clase Implementación

Esta clase va a tener la declaración de variables ‘generales’ y varios métodos:

Worktxt

1. Worktxt: en este método voy a operar sobre los documentos en la carpeta 'Content' para obtener su dirección y su título.

WorkWords

2. WorkWords: en este método voy a recorrer todos los documentos y voy a dejar solo las palabras y números (elimino los caracteres como ‘.’ y ‘,’) y guardo en un diccionario las palabras con su frecuencia y en listas guardo los documentos sin los caracteres y las palabras de todos los documentos sin repetir.

QueryNormalizar

3. QueryNormalizar: en este método voy a eliminar del query todo los caracteres que no sean letras o dígitos para posteriormente hacer su cálculo de TF-IDF, utilizando un método llamado TF e IDF de la query del que voy a obtener un diccionario con las palabras y su valor de TF-IDF asociado.

QueryVectorizado

4. QueryVectorizado: este método lo utilizo para guardar en un 'array' solo los valores de TF-IDF de las palabras del query.

TFxIDF

5 TFxIDF: en este método voy a calcular el TF-IDF de las palabras de los documentos, nótese en el código del calculo que le sumo una cantidad casi nula al contador, esto es una medida para impedir una indefinición dentro del cálculo que deriva en errores de búsqueda (aquí dejo una referencia, para más información, pude consultar las páginas de Wikipedia).

$$TF = \frac{FP}{N}$$

FP- Frecuencia de la palabra (Repetición) N- Número total de palabras.

$$IDF = \log\left(\frac{CD}{ND}\right)$$

CD- Cantidad total de TXT ND- Número de documentos que contiene la palabra

TF-IDF (frecuencia de término – frecuencia inversa de documento): Es una medida numérica que expresa cuán importante es una palabra para un documento en una colección. Esta medida se utiliza a menudo como un factor de ponderación en la recuperación de información y la minería de texto. El TF-IDF aumenta proporcionalmente al número de veces que una palabra aparece en el documento, pero es compensada por la frecuencia de la palabra en la colección de documentos, lo que permite manejar el hecho de que algunas palabras son más frecuentes que otras.

Matriztfxidf

6 Matriztfxidf: con este método voy a formar una matriz en la que voy a almacenar los valores de TF-IDF de las palabras del query en los documentos en los que aparezcan, si no aparecen les pongo valor 0.

SimilitudCos

7 SimilitudCos: en este método voy a calcular lo que se llama ‘similitud del coseno’ utilizando el query vectorizado y la matriz de TF-IDF antes calculados, y voy a obtener el ‘score’ de las palabras de los documentos con respecto al query ingresado, (dejo una referencia extraída de Wikipedia, para más información puede visitar la pagina oficial).

Similitud del coseno: Es una medida de la similitud existente entre dos vectores e un espacio que posee un producto interior con el que se evalúa el valor del coseno del ángulo comprendido entre ellos. Esta función trigonométrica proporciona un valor igual a 1 si en ángulo comprendido es cero, es decir si ambos vectores apuntan a un mismo lugar. Cualquier ángulo existente entre los vectores, el coseno arrojaría un valor inferior a uno. Si los vectores fuesen ortogonales el coseno se anularía, y si apuntasen en sentido contrario su valor sería -1. De esta forma, el valor de esta métrica se encuentra entre -1 y 1, es decir en el intervalo cerrado [-1,1].

Clase Levenshtein:

- Esta clase es enteramente para la creación de las sugerencias del buscador a través del cálculo de distancias de Levenshtein, (debajo referencia extraída de Wikipedia).
- Levenshtein: Es el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra, se usa ampliamente en teoría de la información y ciencias de la computación. Se entiende por operación, bien una inserción, eliminación o la sustitución de un carácter. Este cálculo es muy útil para determinar cuan similares son dos cadenas de palabras y la empleo en mi proyecto con el fin de dar una sugerencia lo más acertada posible en caso de errores ortográficos en el query.

Clase Matriz

Aquí lleve las operaciones con matrices a la programación, ademas utilizo la multiplicación de matrices en el proyecto, ya que multiplico una matriz por un vector,que no es mas que una matriz de una sola columna (fila)

Clase Operadores:

En esta clase se encuentran los métodos con los que voy a definir la modificación sobre el 'score' en base a los operadores ingresados

1. En el caso del operador (!) uso el método 'No Aparece' los documentos que contengan a la palabra con este operador decidí, para que no aparezcan por ningún motivo, hacer su score 0, y de esta forma no presentarlo en los resultados de búsqueda.
2. En el caso del operador (*) uso el método 'Asterisco' decidí aumentar la relevancia de los documentos que contienen esa palabra al multiplicar el Score por la cantidad de (*) ingresados aumentado en 1.

Clase Snippet:

En esta clase voy a tener un único método que será el encargado de construir un fragmento del documento donde se encuentren la(s) palabra(s) del query, este fragmento servirá como una breve introducción al contenido del documento.