Moogle

Informe

Moogle es un proyecto cuyo objetivo es encontrar información referente a un determinado pedido. Para lograrlo es utilizado un modelo muy conocido conseguido desde la riquísima rama de las Matemáticas, el Algebra lineal. Este modelo es bien conocido como TF_IDF, con TF nos referimos a la frecuencia de un término en el documento y con IDF nos referimos a la frecuencia inversa de la cantidad de documentos en los que está contenido un término puntual, es decir mientras más este un término en la base de datos menos relevante será este ej.: palabras como “y”, “o”, los artículos, preposiciones etc... Son palabras no relevantes en una búsqueda mientras que palabras como sustantivos, adjetivos, adverbios son palabras que no siempre están repetidas en un documento, pues estas palabras se les otorga más importancia y por tanto su peso en mayor que 0.

Sobre la estructura del código:

Como se trata de un sitio web, debemos de mostrar resultados lo antes posible por lo que no podemos tardar mucho haciendo cálculos que pudieran hacerse solo una vez.

Estos cálculos son:

***Conseguir las palabras de nuestra base de datos**

***Calcular la matriz de peso de las palabras en la base de dato por cada documento**

***Obtener una copia exacta de cada documento para devolver un pedazo del texto para que el usuario pueda tener una idea de que se trata en el texto**

A momentos de recibir una query se ejecuta en primera instancia una serie de métodos estáticos que darán lugar al indexado de la base dato y al caculo de la matriz de peso luego en el backend el método Query del namespace Moogle.cs que ejecutará una serie de métodos que darán respuesta a la query dada

En primer lugar se calcula el TF_IDF de la query dada inicializando una instancia por cada query. En este método existe un mecanismo que dada una query que no esté textualmente tal y como aparece en la base dato se intenta dar respuesta rectificando posibles faltas de ortografías o intentos de textear una palabra con el fin de otorgarle al usuario una respuesta a dicha query.

De aquí mismo sacamos un fragmento del texto de los documentos a la respuesta conseguida según cada query. Hasta aquí tenemos bien guardado en propiedades no estáticas todo lo conseguido en el proceso de calcular el TF_IDF de cada palabra.

Continuando en el método Query de Moogle.cs nos topamos con una condicional que evalúa si existe respuesta para dicha query de no haberla no se le retornará nada. De haberla evaluamos si existe cercanía entre palabras de la query, si existen estas calculamos dicho operador y almacenamos en una propiedad para luego devolver los documentos y sus snippet (calculados de la cercanía) en orden de menor a mayor.

El buscador también cuenta con otros operadores que son los de exclusión, existencia y prioridad que intentaran evaluar la respuesta de la query dada para tener mayor precisión en los resultados de la búsqueda.

MAS DETALLES:

Evaluar query:

1. **Es necesario instanciar la clase para evitar que al introducir una nueva query en el programa no se almacenen en cola y devuelva resultados insatisfactorios esto lo hacemos en la primera línea que se ejecutará. Una vez en el constructor de la clase TF_IDF evaluamos las recomendaciones del cálculo. De no existir entonces las evaluamos en función de la sugerencia que se devolverá de la clase Sugerencia. De no existir las recomendaciones calculas por la clase TF_IDF se ejecutará la última línea del método Query que preguntará si se quiso decir algo.**
2. **En operadores como el de exclusión solo tenemos que eliminar la estas palabras negadas del cálculo del TF_IDF**
3. **Nuestra query también se evalúa en función de los sinónimos que existen sobre lo que se está buscando, es decir, buscamos que sinónimos hay sobre las palabras de la query y pues, claro como no son palabras de la query ha estos sinónimos se le otorga menos peso de modo que si en un documento existen solo sinónimos pero no existe ninguna palabra de las de la query y en otro existe al menos una palabra de la query pero no existen sinónimos entones este último tiene mayor relevancia.**

Evaluar cercanía:

2. De no ser nulo el cálculo de la clase TF_IDF, es decir de existir resultados de búsqueda entonces se evaluar posible existencia de cercanía. Esta clase básicamente evalúa el operador de cercanía y retorna las palabras que se quiere estén lo más

cercanas posible. Existe un método que ordena estos snippets por cercanía y las que no estén cercanas lo suficiente como para considerar un distancia de a lo más diez palabras entre ellas entonces no se devolverán (por cuestiones de optimización). Quizás la query está mal escrita y por tanto debemos arreglar ese error y sugerir después, esto lo hacemos también, buscando cuales son las palabras en un documento que más se ajusta a las palabras del usuario .Una vez tenido esto y almacenado en propiedades que luego servirán de puente conector entre clases se vuelve al método Query de la clase

Moogle.cs y se devuelve el snippet correspondiente a cada documento . Si no existe snippet para ello entonces devolvemos los resultados de la clase TF_IDF. Chequemos un caso particular, si se pide la cercanía entre dos palabras iguales estas se toman como una palabra puesto que la menor distancia en por ejemplo pero y perro es nada; pues en ese caso solo se devuelve el resultado de la clase TF_IDF.