Informe del Moogle

Magela Cornelio C111

July 23, 2023

ACLARACION: Mis documentos son en ingles

Que es el Moogle?

El proyecto Moogle se basa en un buscador de documentos mediante codigos del lenguaje de C#. El mismo recibe un string Query y devuelve una variable de tipo SearchResult, que contiene un array de SearchItems(titulo, snippet y score del documento) y la sugerencia(que es la similitud de palabras entre el query y el vocabulario).

Como lo hice

Primeramente fui guardando en un string las direcciones de los archivos almacenados en la carpeta content con la clase y el metodo Path.Join (descriccion del metodo) luego cree un array de string con todos los documentos y fui creando una matriz donde fui guardando cada documento. Guarde en una lista de string todas las palabras de todos los documentos creando asi el vocabulario que lo normalize(elimine tildes, caracteres extranos, mayusculas, etc) con el metodo Normalize.Text. Ademas hago exactamente lo mismo con el query, creo una lista de string y la nomalizo igual.
Para guardar los titulos de los documentos utilizo el metodo Title.Of.Each.Documents el cual recibe un array de string de todos los doumentos, y con.Substring(una variable tipo string.LastIndexOf() +1); separo el titulo del libro, el metdodo devuelve un diccionario con cada documento y su titulo. Ahora; empezamos a crear una matriz del mismo tamano de la que ya tenemos creada con todo los documentos para llenarla con el TF y el IDF de cada palabra (el TF(Term of Frecuency) es la cantidad de veces que se repite dicha palabra en cada documento, el IDF es el logaritmo de la division entre la cantiddad total de documentos por la cantidad de docuemntos en los que se repite dicha palabra)

para ello cree el metodo Term.Frecuency (para el TF) que recibe una matriz de documentos y una palabra, este metodo va iterando por la matriz y va comparando cuantas veces se repite la palabra que recibe en los documentos devolviendo asi un double con la cantidad de veces que esta se repite. Y el metodo Inverse.Documents.Frecuency(para el IDF) que recibe exactamente lo mismo, lo que hay un ligero cambio aqui, este metodo devuelve un double con el logaritmo de dividir la cantidad total de documentos(metodo aparte llamado Count.of.Documents que va almacenando en un double la cantidad de filas que tiene la matriz de todos los documentos) por la cantidad de veces que aparece dicha palabra en un documento(metodo aparte tambien, llamado Count.Documents.By.Words que este itera por la matriz donde analiza si la palabra seleccionada aparece en dicho documento) luego multiplicamos el TFy el IDF de cada documento y lo almacenamos en una matriz, creando asi nuestra matriz del TF y el IDF de todos los documentos.
Con el Query me complique un poco pero lo resolvi, para calcular el TF y el IDF del Query estan los metodos TF.Query y IDF.Query pero a diferencia de el de los documentos es que en vez de recibir una matriz, recibe el Query y todo lo demas es exactamente lo mismo, asi igual pasa lo mismo con el IDF.
Nos centramos ahora en la similitud de coseno que se basa en la similitud entre dos vectores (TF IDF de cada documento) mientras mas cerca de 1 este es porque mas similares son, para calcular esto esta el metodo Cosine.Similarity el cual recibe un documento, el Query y su respectiva magnitud, (la cual se obtine con el metodo Vector.Magnitude que recibe la matriz del TF IDF y lo que devuelve es un double con la raiz cuadrada de las suma de los cuadrados de cada vector) una vez ya con todo este metodo devuelve un double
Una vez ya con la similitud de coseno conseguida la guardamos en un diccionario, cada documento con su similitud de coseno(a lo que le vamos a llamar score). Ordenamos el diccionario de mayor a menor porque ya esos documentos son los que entregaremos, luego creamos una lista con los 10 primeros documentos del diccionario ya ordenado. Ya teniendo los documentos a devolver empezamos la tarea del snippet, el snippet es ese pedazo mas importante de texto de cada documento, para lograr conseguir el snippet lo primero es dividir un documento por oraciones(en mi caso quise poner una oracion ya que es mas estetico que poner un parrafo) para dividir el documento utilice el metodo Matrix.Snippet que recibe la lista de documentos a entregar(a partir de ahora nuesta matriz de documentos es la lista de documentos que ya vamos a devolver) este metodo divide cada documento que recibe por oraciones y la almacena en una matriz (un matriz por cada domumento) a esta matriz del snippet le calculo el TF y el IDF de la misma forma que calculamos el de los documentos con los metodos Term.Frecuency

e Inverse.Documents.Frecuency los cuales ya explique anteriormente. Al calcularle el TF y el IDF hacemos la matriz con la multiplicacion de los mismos y a eso le hallamos la similitud de coseno, creando asi un diccionario con cada oracion y su respectivo score(similitud de coseno) ordenamos el diccionario(no encontre un metodo para ordenar un diccionario de mayor a menor lo que hice fue crear una lista le paso los keys y con .Sort los organizo y luego los vuelvo a pasar al diccionario) una vez con el diccionario del snippet y su score ordenados tomamos en una lista de string el snippet que mas score tenga, teniendo asi el snippet del documento.
Antes de devolver tenemos que crear la sugerencia, para la sugerencia uso el metodo de la distancia de Levenstien(similitud de palabras) el cual recibe una palabra del Query y una palabra del vocabulario, este metodo se basa en (explicar esta talla) este metodo devuelve una variable de tipo int con la menor cantidad de combinaciones. Una vez que ya tengamos (....continuar la parte de la sugerencia)
Como ya tenemos todo nada, mas nos queda devolver, con el metodo Result que recibe los titulos, los snippet, los score, de los documentos y ademas los documentos. Este metodo se basa en crear un array de searchItem del tamano de la lista de los documentos a entregar (10).
Para ya devolver creo una variable de tipo SearchResult que necesita el array de los Searchitem y la sugerencia.

Conclusiones

Este proyecto fue mas que un reto para mi ya que fue el primer proyecto al que me enfrento, fue ese cambio de pensamiento e investigacion tales que resultaron tan dificiles y se que a pesar de haberlo entregado fuera de tiempo me dio fuerzas para poder hacerlo mejor y asi conocer mas cada tema. En fin, este fue mi Moogle, mi primer proyecto de programacion.