Informe escrito del proyecto de programación de 1er año de Lic. en Ciencia de la Computación

Nombre: Gabriel Herrera Carrazana

Grupo: C22

Nombre del Proyecto: Moogle!

Enlace de GitHub del proyecto : https://github.com/gabrielherreca/Moogle.git

Durante el desarrollo del proyecto fueron implementados en su totalidad una serie de clases y métodos y fueron modificados otros tantos.

Creados por mi:

-Matrix.cs

-Vector.cs

-LoadDocuments.cs

Modificados:

-Moogle.cs

-SearchItem.cs

LoadDocuments.cs

Aqui se encuentra la clase LoadDocuments donde se cargan y procesan los distintos textos con los que vamos a trabajar . Para esto nos apoyamos de una serie de métodos estáticos

-Load

En este método almacenamos las rutas de los documentos .txt usando el método GetFiles de la clase Directory .Luego almacenamos por cada ruta , el texto correspondiente usando método ReadAllText de la clase File. Finalmente se devuelve una lista de string , donde cada string representa el contenido de un documento .txt.

- Normalize

Aquí usamos el método Split para separae cada documento en palabras . Luego se convierte cada palabra a minúsculas y se quitan los símbolos que estas puedan contener con el método Regex de System.Text.RegularExpressions . Finalmente añadimos cada palabra procesada a una lista de string y es lo que retorna el método .

-DocumetList

Este método devuelve una lista de listas de string , donde cada lista representa un documento y cada string representa cada palabra del documento ya procesada.

- Vocabulary

Este método devuelve un Hashset de string con cada palabra existente y sin repeticiones en nuestra colección de documentos .

-DocumentVocabulary

Devuelve un Hashset de string con cada palabra existente y sin repeticiones en un documento especifico que recibe como parámetro.

Este método sera útil para ser invocado en el método AllDocumentVocabulary() .

- AllDocumentVocabulary

Este método nos permite almacenar en una lista de HashSet , el vocabulario específico todos los documentos con los que se va a trabajar.

6-IDF

Este método recorre cada parlabra del vocabulario obtenida con el método Vocabulary() y se añade a un diccionario con su valor de IDF(inverse document frecuency) correspondiente que se calcula “log10((cantidad total de documentos)/(documentos en que aparece la palabra))”.

– TF

Este método devuelve una lista de diccionarios, cada diccionario representa un documento y contiene cada palabra del texto asociado a su valor de TF(term frecuency) . El TF de una palabra se calcula “(repeticiones de la palabra en el documento) /(cantidad de palabras en el documento))”

-TFOfDoc

Este método se usa para el trabajo con la query. Añade a un diccionario cada palabra de la query con su valor de TF asociado.

-QueryTF IDF

Este método devuelve una diccionario que representa la query . El diccionario contine cada palabra del vocabulario asociada a su valor de TF IDF . Se verifica si la palabra del vocabulario esta en la query . Si esta contenida se calcula su valor de TF IDF a partir de la multiplicación de los valores de la palabra obtenidos en los métodos TFOfDoc() y IDF(). Si no esta contenida se añade la palabra al diccionario con un valor de TF-IDF asociado de 0 directamente , evitando cálculos inecesarios.

-DictionaryTFIDF

En este método creamos una lista de diccionarios , donde cada diccionario representa un documento , y cada diccionario contiene cada palabra del vocabulario asociado a su valor de TF-IDF . Se comprueba en cada documento , si cada palabra del vocabulario esta contenida en el docuemto . Si esta contenida de calcula su valor de TF IDF a partir de la multiplicación de los valores de la palabra obtenidos en los métodos TF() y IDF(). Si no esta contenida se añade la palabra al diccionario con un valor de TF-IDF asociado de 0 directamente , evitando cálculos inecesarios.

-Titles

Este método contine los nombres de los documentos txt. de la carpeta content.

-Titles

Devuelve una lista de string con los nombres de los documentos con los que se va a trabajar.

-Generate Snippets

Genera un snippet a partir de la primera palabra de la query y un documento.

-Lists Of Snippets

Genera un snippet por cada documento existente a partir de la primera palabra de la query.

Vector.cs

En esta clase Vector creamos los objetos de tipo Vector , que reciben en su constructor un diccionario <string,decimal>. De esta manera podemos representar cada documento como un vector , que contiene cada una de las palabras del vocabulario asociado a su valor de TF-IDF.Tambien creamos varios métodos para el trabajo algebraico con los vectores que son los que nos permitirán calcular los Scores.

Métodos:

-DotProduct

Calcula el producto de punto entre dos vectores .Este se obtiene al sumar los productos componente por componente.

-Magnitude

Calcula la magitud de un vector. Este se expresa como la raíz cuadrada de la suma de cada componente elevada al cuadrado.

-MagnitudeProduct

Calcula el producto entre la magnitud de dos vectores.Usa el método Magnitude explicado anteriormente.

-CalculateSimilitudCoseno

Este método calcula la similitud de cosenos entre dos vectores.Se obtiene al calcular el cociente entre el producto de punto entre los dos vectores y el producto de sus magnitudes.

Matriz.cs

Aquí se encuentra la clase Matriz , donde creamos los objetos de tipo matriz. Estos reciben en su contructor una lista de objetos Vector .

Métodos

-ListforMatrix

Este método nos devulve una lista de vectores que nos será útil para instanciar un objeto de tipo Matrix . De esta manera lograríamos tener un objeto que agrupa toda la información que necesitamos de los documentos para el calculo con la query.

-Scores

Este método nos devulve una lista de float con los los valores resultantes de calcular la similitud de cosenos entre cada vector de un obeto tipo Matrix y un vector . Este método lo usaremos posteriormente para obtener los scores entre cada documento y la query.

SearchItem.cs

En esta clase fueron agregados algunos métodos:

SearchItemList

Este método nos devuelve una lista de objetos SearchItem. Cada objeto representa un documeto con su título , su valor de score respecto a la query y un snippet del texto donde aparece la query.Recibe como parametos una lista de float donde estarán los valores de score y un string que representa la query.

SortByScoreDecending

Es un método que ordena una lista de objetos SearchItem de manera descandente según el valor del parámetro Score.

Moogle

En esta clase primeramente instanciamos un objeto DocumentMatrix de tipo Matrix y le pasamos como argumento la lista de objetos Vector que nos devuelve el metodo ListforMatrix de la clase Matrix.

Dentro del método query instasnciamos un objeto Queryvector de tipo Vector y le pasamos como parámetro el método QueryTFIDF de la clase Load Documents . Asi conseguimos llevar la query a su representación vectoral.

También creamos una lista de float llamada Scores para copiar la lista de float que nos devuelve el método Scores de la clase Vector (calcula los valores de similitud de cosenos) , pasándole como parámetros QueryVector y DocumentMatrix.

Por último igualamos el array de SearchItems Items a la lista de objetos SearchItem que nos devuelve SearchItemList (con los parámetros string query y List<float>Scores) luego de ser

ordenada con el método SortByScoreDecending y convertida de lista a array con el métdo ToArray.