Informe escrito del proyecto de programación de 1er año de la carrera de Ciencias de la Computación de la Universidad de La Habana

Arlette Martínez Lemes

C-112

Índice

1. Arquitectura básica del proyecto

Para el correcto desarrollo de este proyecto fue necesario la implementación de nuevas clases y métodos, al igual que la modificación de otros pocos; estos son:

Clases modificadas:	Clases implementadas por mí:
- Moogle.cs	- Document.cs
- Program.cs	- Operations.cs
- Index.razor	- QuerOperations.cs
- Index.razor.css	

2. Flujo de los datos durante la ejecución de la búsqueda

*La mención de métodos será de la forma:
"NombreDeLaClase.NombreDelMétodo"*

2.1. Pre-procesamiento

Una vez sea ejecutado el proyecto, serán efectuados una serie de métodos para el procesamiento de la información con la que se trabajará posteriormente. Primero que todo se accederá al método **"QueryOperations.LoadFiles"**, el cual a partir de los documentos *.txt* encontrados en la carpeta *"Content"* creará un *array* de *Document* con el que ya se podrá trabajar la información. Luego será creada una instancia de la clase **"Operations"**, donde utilizando la técnica **TF-IDF** se procesan los textos evaluando la importancia relativa de cada palabra en cada documento dentro de la colección de documentos.

2.2. Búsqueda

Al presionar el botón *"Buscar"*, es llamado el método **"Moogle.Query"** donde se verifica si la *query* está vacía, devolviendo así un mensaje pidiendo al usuario insertar una consulta. En caso contrario, se comienza a trabajar en la *query* llamando a diversos métodos que se encargarán de normalizar los textos y darles peso a las palabras de la consulta. Luego se crea una instancia de la clase *"SearchItem"*; y a la hora de devolver los resultados que coincidan total o parcialmente con la *query*, estos serán ordenados por el **Score** de manera descendente y con un **Snippet** encontrado por el método **"QueryOperations.Snippet"**, que retornará un pequeño fragmento del texto, que casi siempre contendrá la/s palabra/s buscadas. Además, siempre será retornada una sugerencia de búsqueda que en caso de no encontrarse en el corpus de documentos la palabra deseada, sugerirá otra que será de las sí contenidas en nuestra base de datos, la más parecida lexicográficamente a la búsqueda en cuestión.

3. Funcionalidad de las clases implementadas por mí

*Las propiedades de las clases serán mencionadas de forma: **(.Propiedad)***

3.1. Document

Clase que administra mediante propiedades el nombre **(.Name)**, la dirección **(.Path)** e información de un documento **(.Words)**.

3.2. Operations

Objeto que al ser instanciado en la clase **Program** realiza un conjunto de operacion pre-procesar la información obtenida con anterioridad. Para procesar la información del modo esperado, primero que nada, se accede al método **"Operations.TF"**, con el cual obtendremos la frecuencia de cada palabra en cada documento. Para ello al método se le pasa como parámetro un *array* de *Document* con el texto de cada *txt* siendo sólo números y letras en minúsculas sin tildar. Luego iremos indexando en cada palabra de cada *Document* con ayuda de la propiedad **(.Words)** que contiene esta clase; entonces iremos añadiendo cada palabra a un diccionario (aquí agruparemos todas las palabras distintas de nuestros documentos) ahí se le asociará un índice que será la columna que ocupa la palabra en la matriz donde se relacionan palabras con documentos. Después de calculado el **TF** (Term Frecuency) se accederá al método **"Operations.IDF"** donde se le pasa como parámetro una lista de *array* de *double* (matriz TF) y se calcula el **IDF** (Inverse Document Frecuency) de cada palabra utilizando la expresión matemática:

$$IDF = \log(\frac{1+N}{n_p}) \tag{1}$$

Donde N representa el número total de documentos y n_p el número de documentos que contienen la palabra en cuestion. Así finalmente se accede al método **"Operations.TF_IDF"** donde se multiplicará la matriz de **TF** por el *array* de **IDF**, obteniendo la matriz de **TF-IDF**.

3.3. QueryOperations

Clase que contiene múltiples métodos usados mayormente en el trabajo con la *query*. Los métodos encontrados en esta clase son; **"QueryOperations.LoadFiles"** y **"QueryOperations.Normalize"** que serán utilizados a priori de la inserción de la *query* y los cuales son fundamentales en la obtención limpia de la información base. Luego estarán los métodos **"QueryOperations.QueryNormalize"**, **"QueryOperations.Symbols"**, **"QueryOperations.QueryTF"** y

"QueryOperations.QueryTF_IDF" que se encargarán de dar valor a las palabras de la *query*, contemplando el máximo valor para aquellas palabras que consten en su estructura del símbolo **$**. También aquí se verá **"QueryOperations.CosineSimilarity"** que acompañado de **"QueryOperations.Norms"** y **"QueryOperations.QueryNorms"**, mide la similitud entre dos vectores o documentos. Finalmente **"QueryOperations.DescendingOrder"** que organiza de forma descendente los documentos que coinciden total o parcialmente con la consulta dada; **"QueryOperations.Snippet"** que extrae de cada documento devuelto un pequeño fragmento en el que aparece casi siempre la/s palabra/s que coincidieron. Y **"QueryOperations.SimilarSuggestion"** que auxiliado por **"QueryOperations.LevenshteinDistance"** hace que según la consulta echa, sea posible (en caso de no encontrar ninguna coincidencia) sugerir una consulta que siendo esta la más parecida a la anteriormente echa, además aparezca **100 %** en la base de datos.

4. Carpeta AlgebraContent y sus clases

En esta sección hablaremos de las clases implementadas para cubrir las exigencias de la materia de **Álgebra I**.

4.1. Matrix

Objeto que al ser instanciado consta de varios métodos que permiten al usuario realizar operaciones con matrices. Estos métodos son:

- **"Matrix.MatrixAddition"**: Suma dos matrices.
- **"Matrix.MatrixMultiplication"**: Multiplica dos matrices.
- **"Matrix.MultiplicationForNumber"**: Multiplica una matriz por un escalar.
- **"Matrix.MultiplicationForVector"**: Multiplica un obejto Matrix por un objeto Vector.
- **"Matrix.MatrixSubtraction"**: Halla la diferencia de dos matrices.
- **"Matrix.MatrixTranspose"**: Halla la traspuesta de una matriz.

4.2. Vector

Objeto que al crear una instancia nos brinda la posibilidad de usar sus métodos con el fin de realizar operaciones algebraicas con vectores. E aquí los métodos:

- **"Vector.VectorAddition"**: Suma dos vectores.

- **"Vector.VectorSubtraction"**: Halla la direncia de dos vectores.
- **"Vector.VectorMultiplication"**: Multiplica dos vectores.
- **"Vector.MultiplicationForNumber"**: Multiplica un vector por un escalar.

5. Funcionalidad extra implementada

En este buscador el símbolo **$** tiene como utilidad la de dar mayor importancia a los documentos que coincidan con la/s palabra/s a la/s que se le/s agregó tal símbolo. Para que esta funcionalidad tenga efecto la palabra deberá estar bien escrita, ya que no importa en qué parte de la palabra se encuentre el símbolo, igual lo reconocerá, pero sólo dará los resultados esperados si la palabra está correcta al quitar el símbolo.

Ejemplos:

Figura 1: Búsqueda sin $

Figura 2: Búsqueda con $

Figura 3: Búsqueda incorrecta