Presentacion

Dario Lopez Falcon

Julio, 2023

1. Introducción

Este documento fue creado con el fin de explicar como funciona el codigo de Moogle, dando a conocer sus principales clases y las propiedades y metodos que contien cada una de estas. Tambien hablaremos sobre la similitud de Cosenos y de los vectores TF y IDF ; ya que el algoritmo de busqueda de Moogle esta basado en estos.

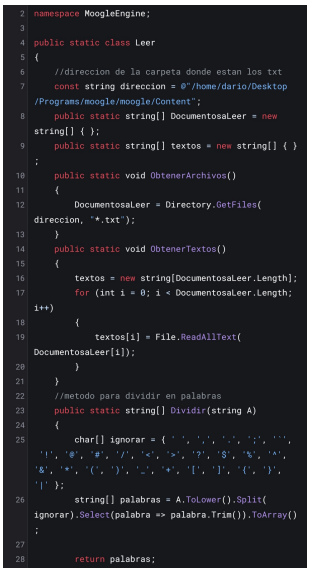
2. Clases:

Es importante destacar que el codigo esta comentado en su mayoria, por lo que si quiere profundizar mas en como funciona Moogle! les recomiendo que lq hechen un vistaso.

2.1. Class Leer

El objetivo de esta clase como bien dice el nobre es leer todos los archivos .txt que se encuemtran en la base de datos , y tambien guardar el nombre y el contenido de cada archivo para poder ser usados posteriormente.

Para poder realizar estas funciones dicha clase cuenta con las propiedades y metodos que se muestran en la siguiente imagen:



```
namespace MoogleEngine;

public static class Leer
{
    //direccion de la carpeta donde estan los txt
    const string direccion = @"/home/dario/Desktop
/Programs/moogle/moogle/Content";
    public static string[] DocumentosaLeer = new
string[] { };
    public static string[] textos = new string[] { }
;
    public static void ObtenerArchivos()
    {
        DocumentosaLeer = Directory.GetFiles(
direccion, "*.txt");
    }
    public static void ObtenerTextos()
    {
        textos = new string[DocumentosaLeer.Length];
        for (int i = 0; i < DocumentosaLeer.Length;
i++)
        {
            textos[i] = File.ReadAllText(
DocumentosaLeer[i]);
        }
    }
    //metodo para dividir en palabras
    public static string[] Dividir(string A)
    {
        char[] ignorar = { ' ', ',', '.', ';', ':', '`',
'(', '@', '#', '/', '<', '>', '?', '$', '%', '^',
'&', '*', '(', ')', '_', '+', '[', ']', '{', '}',
'|' };
        string[] palabras = A.ToLower().Split(
ignorar).Select(palabra => palabra.Trim()).ToArray()
;

        return palabras;
```

2.2. Clas Documento

Por la forma en que diseñe Moogle! cada archivo de la base de datos se convertira en cun documento, se hizo indispensable definir esta clase y darle las propiedades y metodos necesarios para la construccionde cada documento construccion de cada documento y posteriormente realizar la busqueda.

```
public class Documento
{
    public string titulo;
    public string texto;
    public string[] TodasLasPalabras;
    public Dictionary<string, int> palabras;
    public Dictionary<string, float> TF_IDF;
    public float score = new float();
    public void LlenarPalabras()
    {
        for (int i = 0; i < TodasLasPalabras.Length;
 i++)
        {
            if (palabras.ContainsKey(
TodasLasPalabras[i]))
            {
                palabras[TodasLasPalabras[i]]++;
            }
            else
            {
                palabras.Add(TodasLasPalabras[i], 1)
;
            }
        }
    }
    public void LlenarTF_IDF()
    {

        foreach (var A in palabras)
        {
            TF_IDF.Add(A.Key, (float)palabras[A.Key]
 / TodasLasPalabras.Length * Moogle.IDF[A.Key]);
        }

    }
```

Aqui ademas les dejo el constructor de la clase.

```
    //constructor
    public Documento(string titulo, string texto)
    {
        this.titulo = titulo;
        this.texto = texto;
        this.TodasLasPalabras = Leer.Dividir(texto);
        this.palabras = new Dictionary<string, int>(
);
        this.TF_IDF = new Dictionary<string, float>(
);
    }
```

2.3. Class Query

De manera muy similar a como es la clase Documento crearemos una nueva clase llamada Query.

```
public class Query
{
    string texto;
    string[] TodasLasPalabras;
    string[] TodasLasPalabrasConCaracteres;
    public Dictionary<string, int> palabras;
    public List<string> TienenQueEstar = new List<
string>();
    public List<string> NoPuedenEstar = new List<
string>();
```

En cada busqueda convertiremos la consulta realizada en una Query .(Si desea ver los el construtor mire el codigo).

Esta clase es de vital importancia , pues en ella al igual que en cada uno de los documentos se almacena un diccionario llamado TF*IDF que sera nuestro vector TF*IDF

(del cual hablaremos mas adelante) al cuales se le realiza la similitud de coseno con cada uno de los vectores TF*IDF de los documentos para asi saber que tan relevante es un documento con respecto a la Query.

En esta clase tambien podemos encontras mas metodos que son empleados para realizar busquedas mar personalizadas, a lo cual si desea saber en que consiste le recomiendo ver la presentacion del proyecto y posteriormente analiasar el codigo.

2.4. Class Moogle

Finalmente llegamos a la clase Moogle, y digo finalmente ya que es en esta clase donde se almacenan todos los metodos para contruir los documentos y la Query , ademas de los metodos necesarios para realizar la similitud de cosenos y organizar la salidaa. Pero como si todo esto fuera poco es donde se llaman y ejecutan dichos metodos y los de las demas clases, por lo qu e podriamos decir que es el cuerpo de noestro proyecto. Habria quedado mas organizado creando otra clase que almacenara todos los metodos y en esta solo sean llamados para ejecutarse.

A continuacion les dejo unas imagenes con todo lo que contiene la clase y como ya he dicho si quiere ver en porofundidad como funciona cada metodo y el orden que sigo para que se ejecuten les recomiendo ver el codigo que esta comentado.

```
1  namespace MoogleEngine;
2
3  public static class Moogle
4  {
5      public static int cantidadDeDocumentos = 0;
6      public static Documento[] documentos = new
    Documento[] { };
7      public static Dictionary<string, float> IDF =
   new Dictionary<string, float>();
8      static Moogle()
9      {
10         ConstruirDocumentos();
11         LlenarIDF();
12         CrearTF_IDF();
13     }
```

Class Matrix

POr la forma en que diseñe el algoritmo de busqueda no empleo esta clase en ningun momento,pero esta forma parate del proyecto ya que esta representa una abstraccion de lo que seria una matriz en las matematicas, por lo que en esta clase podemos encotrar metodos que definen la suma ,el producto ,etc entre matices.

```
public class Matriz
{
    double[,]matriz;
    public Matriz Suma(Matriz A,Matriz B)
    {
        double[,] suma=new double[A.matriz.
GetLongLength(0) ,A.matriz.GetLength(1)];

        for(int i=0;i<A.matriz.GetLength(0);i++)
        {
            for(int j=0;j<A.matriz.GetLength(1);j++)
            {
                suma[i,j]=A.matriz[i,j]+B.matriz[i,j
];
            }
        }
        return new Matriz(suma);
    }
```

```
    public Matriz Producto(Matriz A,Matriz B)
    {
       double[,]producto=new double[A.matriz.
GetLength(0),B.matriz.GetLength(1)];

        for(int i=0;i<A.matriz.GetLength(0);i++)
        {
           for(int j=0;j<B.matriz.GetLength(1);j++)
           {
                for(int k=0;k<A.matriz.GetLength(1);
k++)
                {
                    producto[i,j]+=A.matriz[i,k]*B.
matriz[k,j];
                }
           }

        }
       return new Matriz(producto);
    }
```

```
37      public Matriz ProductoPorEscalar(Matriz A, float
    b)
38      {
39          double[,]Producto=new double[A.matriz.
    GetLength(0),A.matriz.GetLength(1)];
40
41          for(int i=0;i<A.matriz.GetLength(0);i++)
42          {
43              for(int j=0;j<A.matriz.GetLength(1);j++)
44              {
45                  Producto[i,j]=A.matriz[i,j]*b;
46              }
47          }
48          return new Matriz(Producto);
49      }
50      public Matriz(double[,]a)
51      {
52          this.matriz=a;
53      }
```