MoogleServer:

NOTA: Busca al presionar Enter

En MoogleServer/Program.cs se llama a la función que carga todos los datos necesarios:

```
MoogleEngine.LeerDocs.GetData();
app.Run();
```

LeerDocs:

La clase estática "**LeerDocs**" contiene la función "**GetData()**" que utiliza la clase "Directory" contenida en la biblioteca de C# "System.IO" para obtener un array con la dirección de todos los archivos txt y luego lo recorro leyendo su contenido y creando un "*Vector*" por cada documento:

```
string[] files = Directory.GetFiles(path, "*.txt");      You, la semana pasada • Sin c
foreach(string file in files){
    string text = "."+File.ReadAllText(file).ToLower()+".";
    //Creo una lista con todas las palabras del texto
    List<string> listWords = (Utils.MakeList(text).ToList());
    //Creo un Vector por cada documento (Al crearse se guarda en mi matriz de docs)
    Vector v = new Vector(
        file,
        listWords
    );
}
```

Vector:

La clase *Vector* tiene las siguientes propiedades:

- **TFIDF** (Dictionary<string, double>)
- **Path** (string)
- **Words** (List<string>)

TFIDF: Al crear un Vector se llama a la función "CountWords()" que llena este diccionario con la cantidad de veces que se repite cada palabra en el documento al que está asociado.

Path: Al crear el vector de los documentos se le asigna la dirección del documento en la computadora (desde el constructor).

Words: Contiene una lista con todas las palabras del documento.

NOTA: Al crear un vector, éste se guarda en la clase *Matriz*.

Funciones de la clase Vector:

- **CountWords()**
- **GetName()**
- **ProdEscalar(Vector v)**

CountWords(): Llena el Dictionary **TFIDF** con la cantidad de veces que se repite cada palabra. Y llena el Diccionario<string, int> **IDF** de la clase "*Matriz*" con la cantidad de documentos en los que aparece cada palabra del texto.

(Ej: Si la palabra "hola" aparece en 5 docs, un elemento de *Matriz***.IDF** sería <hola, 5>).

GetName(): Obtiene el nombre del documento a partir de su **Path.**

ProdEscalar(Vector v): Se llama al realizar una query, (*queryVector.ProdEscalar(anotherVector)*),

1. Recorre las palabras de la query
2. Calcula el TF-IDF de cada palabra de la query y lo guarda el Dictionary **TFIDF**
3. Si el vector "v" que recibe por parámetro contiene la palabra procede a calcular el TF-IDF de dicha palabra en el vector
4. Luego multiplica los TF-IDF de la palabra en cada Vector para así hallar el producto escalar entre ambos vectores

Matriz:

La clase estática *Matriz* tiene las siguientes propiedades:

- **MatrizVectores** (List<*Vector*>)
- **IDF** (Dictionary<string, int>)

MatrizVectores: Cada vez que se crea un vector se agrega a esta lista de vectores (Contiene todos los vectores de todos los documentos de la carpeta Content).

IDF: Al llamarse la función **CountWords()** de la clase *Vector* éste dictionary se llena con la cantidad de documentos en los que aparece cada palabra del texto.

Funciones de la clase Matriz:

- **Add(**Vector v**)**
- **CalculateIDF(**string w**)**
- **CalculateTF(**Vector v, string w**)**

Add(Vector v**):** Agrega el vector "v" a **MatrizVectores.**

CalculateIDF(string w**):** Calcula el IDF de una palabra. Se usa al hacer el query por lo que ya **IDF** contiene la cantidad de documentos en los que aparece cada palabra del texto.

CalculateTF(Vector v, string w**):** Calcula el IDF de una palabra. Se usa al hacer el query por lo que ya v.**TFIDF** contiene la cantidad de veces que aparece la palabra en el texto.

Search:

La función **Search()** ubicada en la clase *Moogle* retorna un array de **SearchItem**:

1. **Calcula el score** por cada vector y solo lo muestra si es mayor a 10^-6 (Producto escalar entre el query y el vector)
2. Guarda la posición de la palabra con más TF-IDF del query que esté en el documento
3. **Guarda el snippet** dado por un fragmento del texto que contenga dicha palabra
   (Desde punto (".") anterior a 50 caracteres antes de la palabra hasta el siguiente punto después de 100 caracteres)
   Para ello se usan las funciones **NextDot()** y **PrevDot()** de mi clase *Utils.*
4. Guarda en una **Lista** de **SearchItem** los datos del vector (nombre, snippet, score)
5. Cuando termina de recorrer los vectores ordena la **Lista** de **SearchItem** según su score, la convierte en Array y la devuelve.

Utils:

La clase estática *Utils* tiene las siguientes funciones:

- **MakeList(**string text)
- **PrevDot(**int pos, string text**)**
- **NextDot(**int pos, string text**)**

MakeList(string text**):** Convierte un string (text) en una **List<string>** utilizando **ToLower()** y **Split()** sin caracteres especiales.

PrevDot(int pos, string text**):** Utilizando recursividad busca el punto anterior a la posición "pos" en el texto "text".

NextDot(int pos, string text**):** Utiliza **IndexOf(**'.'**)** para saber la posición del siguiente punto en el texto "text" luego de la posición "pos".