Glenda Natalí Ríos Rodríguez

El usuario debe insertar una consulta y aparecerán en pantalla los nombres de los documentos que tengan alguna similitud con la consulta ordenados de forma decreciente.

Insertemos el siguiente concepto:

Palabras raras: se les llamará palabras raras a aquellas palabras que aparezcan en una menor cantidad de documentos.

Nótese que mientras más rara sea una palabra aparecerá en una menor cantidad de documentos por lo que la palabra tendrá una mayor relevancia a la hora de calcular la similitud entre la consulta y los Documentos guardados en la carpeta “Content” en “moogle”.

Si una palabra aparece en casi todos los Documentos entonces esta palabra tendrá un valor de relevancia despreciable y no tendrá importancia a la hora de calcular la similitud entre la consulta y los documentos.

¿Qué puede hacer mi Moogle?

~Si su consulta es vacía aparecerá en pantalla el texto “No existen Documentos relacionados con su Consulta”.

~Si su consulta no contiene palabras raras(o sea, que las palabras que introduzca el usuario están en todos los documentos Ej: the) entonces aparecerá en pantalla “No existen documentos relacionados con su Consulta”.

~Si en su consulta aparece(n) alguna(s) palabra(s) inicializada(s) con el operador ‘!’ (o sea , !palabra) entonces cualquier Documento que contenga dicha(s) palabra(s) no aparecerá en pantalla.

Ejemplo:

Si usted introduce la consulta : NORA , aparecerá en pantalla el documento cuyo título es A Doll's House.txt. Mientras que si usted introduce la consulta !NORA, no aparecerán Documentos en pantalla.

~Si en su consulta aparece(n) alguna(s) palabra(s) inicializada con el operador '^' (o sea , ^palabra) entonces los Documentos que salgan en pantalla contendrán obligatoriamente dicha(s) palabra(s).

Ejemplo: si escribe la consulta: "^NORA musqueteers" aparecerá en pantalla únicamente el documento cuyo título es A Doll's House.txt , mientras que si usted introduce la frase "NORA musqueteers" saldrán en pantalla los Documentos cuyos títulos son A Doll's House.txt , The History of Tom Jones.txt.

Nota: obviamente ninguna palabra puede contener ambos operadores '^' '!' a la vez (o sea , !^palabra).

~Si en su consulta aparece(n) alguna(s) palabra(s) inicializadas con el operador '*' (puede estar cuantas veces el usuario quiera) (o sea ,*…*palabra) entonces se le dará más relevancia a dicha palabra a la hora de la búsqueda (pd: esto no implica que los documentos con mayor similitud contienen necesariamente dicha(s) palabra(s)).

Ejemplo: si escribe la consulta "Silver Buttler" aparecerá como Documento más similar "The Strange Case of Dr. Jekyll and Mr.Hyde". Luego si usted inserta la consulta "*Silver Butler" aparecerá como Documento más similar "The Silver Key".

¿Cómo funciona mi Moogle?

Moogle! es una aplicación *totalmente original* cuyo propósito es buscar inteligentemente un texto en un conjunto de documentos.

Es una aplicación web, desarrollada con tecnología .NET Core 6.0, específicamente usando Blazor como *framework* web para la interfaz gráfica, y en el lenguaje C#.

La aplicación está dividida en dos componentes fundamentales:

- `MoogleServer` es un servidor web que renderiza la interfaz gráfica y sirve los resultados.

- `MoogleEngine

En esta última es donde hemos implementado el algoritmo de búsqueda de nuestro moogle.

Tenemos un método `Moogle.Query` que está en la clase `Moogle` del proyecto `MoogleEngine`.Este método devuelve un objeto de tipo `SearchResult`. Este objeto contiene los resultados de la búsqueda realizada por el usuario, que viene en un parámetro de tipo `string` llamado `query`.

Previamente antes de lanzar un resultado debimos precalcular algunas cosas que iban a relacionar que tan similar era la 'query' con un documento , los resultados aparecerían ordenados por cercanía .

Primeramente en la carpeta Content que está contenida en 'moogle' hemos copiado archivos en formato txt que son los que utilizaremos en la búsqueda.

Hemos creado una clase pública Documento donde creamos variables de tipo documento , estos documentos tendrán varias propiedades . Estas propiedades son:

```
    public string Titulo;
    public string[] Palabras;
    public Dictionary<string, int> ContadorDePalabras;
    public int CantidadDePalabras;
```

Al constructor de esta clase es necesario pasarle los siguientes parámetros

```
public Documento(string titulo, string texto)
```

¿Donde obtenemos dichos parámetros?

Existe una clase pública LeerDocumentos donde en su constructor él leera todos los archivos que se encuentran en una dirección que nosotros insertamos , extraerá el string título y el string texto que necesitamos para construir cada Documento.

Es necesario dar a conocer que existe un método muy utilizado para calcular que tan relevate es una palabra en un conjunto de documentos que contienen muchísimas palabras .

Este método se le conoce como TFIDF.

El TF no es mas que por decirlo así la frecuencia con la que aparece una palabra en un mismo archivo , y la formula para calcularlo es TF(palabra)=Cantidad de veces que aparece la palabra en el texto/ Cantidad de palabras del texto.

El IDF es quien calcula qué tan rara es una palabra en un conjunto de palabras en un conjunto de textos. La fórmula para calcularlo es IDF(palabra)=ln(Cantidad de textos/ cantidad de textos donde aparece la palabra). Por lo tanto , nótese que si una palabra aparece en todos los textos entonces el IDF de esta palabra es 0.

Luego el TFIDF(palabra)=TF(palabra)* IDF(palabra).

En el proyecto existe una clase Vector , donde se crean variables de tipo vector que contienen las siguientes propiedades y a cuyo constructor debes introducirle una variable de tipo Documento.

```
    public Dictionary<string, double> TFIDF;
    public double Norma;

    public Vector(Documento documento)
```

Tambien existe una clase Cuerpo donde introducimos un Documentos[] y que tiene como propiedades

```
  public Dictionary<string, double> IDF;
  public Vector[] Vectores;
```

PD: el IDF se precalcula en el Cuerpo pues es único para cada palabra.

Luego en la clase Moogle vamos a introducir como propiedades un Cuerpo y además un []Documento.

Una vez insertada la query , que entra siendo inicialmente un string . Esa query la convertimos en un vector . En la clase Vector existe una sobrecarga que te construye un vector luego de pasar un string.

Luego procesamos la consulta usando el método ProcesarConsulta donde se le pasa el Vector query, este método te calcula las cosas necesarias de la consulta para luego llevar a cabo el método Similitud de Coseno.

Este método te calcula que tan cercanos son dos documentos y devuelve un valor entre 0_1. Mientras mayor sea ese valor mas similitud habrá entre la consulta y el resultado y por lo tanto será mas probable que sea ese el documento que el usuario está buscando.

Finalmente se ordenarán en un orden decreciente los documentos y aparecerán en pantalla el titulo del documento y un pedazo de texto donde aparezcan palabras de la query .