Informe Escrito

Proyecto de Programación de 1er Año. Licenciatura en Ciencia de la Computación

Nombre: Jorge Alejandro Echevarría Brunet
Grupo: C112

1.- Arquitectura básica del Proyecto:
Durante el desarrollo del proyecto fueron creadas un total de dos clases:
- Reader.cs
- Initialize.cs

Así también como tuvo lugar la edición de otras, estas son:
- Program.cs
- Moogle.cs
- Index.razor

2.- Flujo de datos durante la ejecución de la búsqueda:

Preprocesamiento:

Una vez iniciada la ejecución del proyecto se realiza un llamado del método **Feed** de la clase **Initialize.cs** perteneciente al namespace **MoogleEngine,** en la clase **Program.cs**. Este método tiene como función principal hacer un llamado a su vez de todos aquellos métodos de la clase **Reader.cs** que trabajan directamente con los diccionarios creados en la clase **Initialize.cs,** donde se encuentran almacenados todos los documentos de extensión "**.txt"** de la ruta: **"..\Content"**, así como las palabras asociadas a cada uno de sus textos y su Peso.
Los diccionarios a los cuales hacemos referencia son:
- **IDF**
- **Files**
- **Texts**

Estructura interna de cada diccionario:

IDF: Contiene todas las palabras de todos los textos y su Inverse Document Frequency (IDF) asociada, que no es más que una norma para determinar la frecuencia inversa de un término dado en todos los documentos donde aparezca el mismo. Esta norma se obtiene a partir de la siguiente fórmula matemática:

$$\mathbf{IDF_i = Log10\ (N/n_i)}$$

Donde:

IDF_i: Representa el IDF del término **i.**
N: Representa la cantidad total de documentos.
n_i: Representa la cantidad de documentos donde aparece el término **i.**

Files: Contiene cada documento de extensión **".txt''**, así como las palabras asociadas a cada texto de los mismos y su **Weight**. El cálculo de esta norma está dado por la siguiente ecuación matemática:

$W_{i-j} = freq_{i-j}/max_l freq_{l-j} \times IDF_i$

Donde:

W_{i-j}: Representa el peso del término **i** en el documento **j.**

$freq_{i-j}$: Representa la frecuencia del término **i** en el documento **j.**

$max_l freq_{l-j}$: Representa el término **l** de mayor frecuencia en el documento **j**.

Texts: Contiene cada documento de extensión **".txt"** y asociado a cada uno su respectivo texto.

Búsqueda:

Al introducir la búsqueda deseada y hacer click en el botón **"Buscar"**. En caso de ser una búsqueda (o **query**) **NO** vacía, se aplicará una normalización a la misma, eliminando todos los caracteres especiales no relevantes para el procesamiento, utilizando el método **Clean** de la clase **Reader.cs**. Seguidamente se ubicará cada palabra normalizada de nuestra **query** en el diccionario **QueryWeight** de la clase **Moogle.cs** mediante el método **FeedQW** de la clase **Reader.cs**, donde a cada palabra se le asociará su peso.

El cálculo del peso de un término en una **query** se realiza mediante la siguiente fórmula:

$$W_{i-q}=(a+(1-a)freq_{i-q}/max_l freq_{l-q})xIDF_i$$

Donde:

W_{i-q}: Representa el peso del término **i** en la **query.**

$freq_{i-q}$: Representa la frecuencia del término **i** en la **query**.

$max_l freq_{l-q}$: Representa el término **l** de mayor frecuencia en la **query.**

Acto seguido se procede a buscar entre los documentos, aquellos que satisfagan total, o al menos parcialmente nuestra búsqueda.

Para ello se emplea la fórmula de Similitud de coseno:

$$\mathbf{Rsim(d_j,q)} = \frac{\sum_{i=1}^{n} Wi,j * Wi,q}{\sqrt{\sum_{i=1}^{n}(Wi,j)^2} * \sqrt{\sum_{j=1}^{n}(Wi,j)^2}}$$

Donde:

$Rsim(d_j,q)$: Representa la **puntuación** o **Score** del documento j con respecto a la **query**.

Wi,j: Representa el peso del término **i** en el documento **j**.

Wi,q: Representa el peso del término **i** en la **query**.

Si en alguna palabra de nuestra búsqueda apareciera algún/unos de los operadores **'~'**, **'!'**, **'^'**,**'*'**, estos influirán directamente en el puntaje final de los documentos.

Haciendo uso del método **OPS** de la clase **Reader.cs** se reconocerán dichos operadores y se procederá de la siguiente forma en caso de existir alguno/unos de los mencionados:

-Operador de cercanía **'~'**: Si la palabra que lo posee es la primera de la búsqueda, este será ignorado puesto que se tienen en cuenta tanto la palabra precedida del operador de cercanía, como su antecesora.

En cualquier otro caso se procederá al diccionario **Texts** de la clase **Initialize.cs** y se determinará la distancia entre ambas palabras mediante el método **Distance** de la clase **Reader.cs**. De ser esta menor o igual a 10 palabras, su puntaje será considerablemente alto, en otro caso se tendrá en cuenta al mostrar los resultados finales.

-Operador de NO Aparición **‘!’**: Si alguna palabra presenta este operador, todo documento que la contenga será automáticamente descartado de los resultados finales.

-Operador de Aparición **‘^’**: Si alguna palabra presenta este operador, todo documento que no la contenga será automáticamente descartado de los resultados finales.

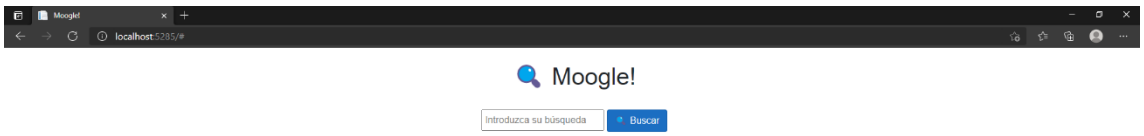
-Operador de Importancia **‘*’**: Si alguna palabra presenta este operador, todo documento que la contenga adquirirá mayor relevancia en dependencia de la cantidad de operadores **‘*’** que la precedan para llevar este conteo se utilizó el método **Charcounter** de la clase **Reader.cs**.

Cada documento poseerá mayor o menor relevancia con respecto a nuestra búsqueda y será ubicado en el diccionario **Score** de la clase **Moogle.cs** mediante el método **FeedScore** de la clase **Reader.cs**, donde se le asociará su respectiva puntuación. Aquellos con una puntuación igual a 0 serán descartados y ordenaremos de forma descendente los restantes, tomando solamente los 3 primeros de mayor puntaje (pudiendo ser menos si fuese el caso).

Resultados:
Una vez ubicados los 3 documentos de mayor puntaje, serán mostrados en pantalla en orden de relevancia, precedidos con la línea de texto asociada a los mismos más afín a nuestra búsqueda. Para ello se utilizó el método **Snippet** de la clase **Reader.cs**.
En caso de realizar una búsqueda donde las coincidencias sean nulas, se ofrecerá en pantalla la/las palabra/palabras que más se asemeje/asemejen a nuestra búsqueda. Para ello se utilizaron los métodos **EditDistance,** en el cual está implementado el llamado **”Algoritmo de Levenshtein”**, el cual determina la mínima cantidad de ediciones a realizar para convertir una palabra en otra y **Suggestion,** el cual determina en base a los resultados del método **EditDistance,** cuál de todas las palabras de mi diccionario **IDF** es la que más se ´´asemeja´´ a mi búsqueda.

Manual de Usuario:

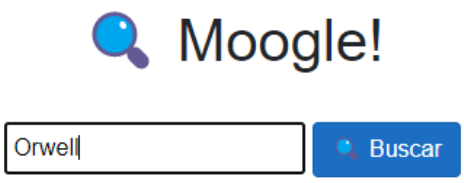
<u>Al Acceder a la página se muestra de la siguiente forma:</u>



<u>Cómo realizar una búsqueda con Moogle!?:</u>

1.- Escribir en la caja de texto la búsqueda a realizar.

2.- Hacer click en el botón “Buscar”



3.- A continuación se mostrarán los documentos de mayor puntaje y una línea de texto que contendrá al menos una palabra de la búsqueda realizada.

En caso de que se haya cometido algún error de escritura o no se encuentren coincidencias entre los documentos, se mostrará siempre una sugerencia con la posible búsqueda a realizar acorde a los documentos existentes.

Operadores:
El usuario puede hacer uso de operadores para facilitar su búsqueda. Estos son:

- Cercanía **(~):** Si es utilizado entre dos palabras o más, cualquier documento que las contenga y se encuentren relativamente cerca, tendrá mayor relevancia que los demás.

- No Aparición **(!):** Si precede alguna palabra, **no** mostrará aquellos documentos en los que la misma esté presente.

- Aparición exclusiva **(^)**: Si precede alguna palabra, mostrará aquellos documentos en los que la misma esté presente.

- Importancia **(*):** Si precede alguna palabra, documentos que la contengan poseerán una mayor relevancia.