Proyecto de Programación:
Licenciatura en Ciencia de la Computación.

Jorge Alejandro Echevarría Brunet.

Facultad de Matemática y Computación.

Abstract

Moogle! es una aplicación web *totalmente original* desarrollada con tecnología .NET Core 7.0, específicamente usando Blazor como *framework* para la interfaz gráfica, y en el lenguaje C#, cuyo propósito es buscar inteligentemente un texto en un conjunto de documentos.

Modelos de Recuperación de Información

¿Qué es la Recuperación de Información?

La recuperación de información es un proceso de comunicación. Es un medio por el que los usuarios de un sistema o servicio de información pueden encontrar los documentos, registros, imágenes gráficas, o registros de sonido que satisfagan sus necesidades o intereses.

Un Modelo de Recuperación de Información (MRI) es el responsable de encontrar información relevante, la cual está presente en un registro documental que debe ser almacenado, representado, analizado (manipulado) y mantenido.

Existen tres Modelos clásicos de Recuperación de Información:

- Booleano: Este modelo tradicionalmente utilizado en bases de datos establece una relación booleana entre los términos y las consultas de búsqueda. Los documentos son vistos como conjuntos de términos. La relevancia se basa en la coincidencia exacta entre los términos de la consulta y los términos en los documentos.
- Vectorial: Este modelo representa los documentos y las consultas como vectores en un espacio multidimensional, donde cada término corresponde a una dimensión. La relevancia se determina mediante cálculos de similitud, como la similitud del coseno, entre los vectores de los documentos y las consultas.
- Probabilístico: Estos modelos utilizan probabilidades para determinar la relevancia de los documentos. El modelo de lenguaje de probabilidad, por ejemplo, considera la probabilidad de que un documento genere una consulta o que una consulta genere un documento, lo que permite calcular la probabilidad de relevancia.

¿Cuál fue modelo fue utilizado para el desarrollo de Moogle!?

El **Modelo de Espacio Vectorial Ponderado** es una variante del modelo vectorial, en la que se asignan pesos a los términos en función de su importancia relativa. Los pesos comúnmente utilizados son los basados en TF-IDF (frecuencia de término-inversa de frecuencia de documento), que tienen en cuenta tanto la frecuencia del término en el documento como en el conjunto de documentos.

1 Arquitectura básica del Proyecto

Durante el desarrollo de la aplicación fueron creadas las clases:

- Reader
- TF-IDF
- Initialize
- Search
- Operators
- Snippet

Así también como tuvo lugar la edición de otros archivos como fueron:

- Program.cs
- Moogle.cs
- Index.razor

2 Flujo de Datos

2.1 Preprocesamiento:

Una vez iniciada la aplicación, se realiza un llamado al método **Feed** de la clase **Initialize**, perteneciente al namespace **MoogleEngine** en la clase **Program**, la cual pertenece al namespace **MoogleServer**. Este método tiene como función principal hacer un llamado a todos los métodos de las clases **Reader** y **TF-IDF**, que trabajan directamente con los diccionarios creados en la clase **Initialize**.

Los diccionarios a los que hacemos referencia son:

- Files
- IDF
- Texts

Estructura interna de cada diccionario:

- Files: Contiene todos los documentos de extensión ".txt", las palabras asociadas a cada texto y el peso de los mismos. El cálculo de esta norma está dado por la siguiente ecuación matemática:

$$W_{ij} = \frac{freq_{ij}}{max_l freq_{lj}} IDF_i$$

Donde:

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

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

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

. IDF_i: Representa la IDF del término i.

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

$$IDF_i = \lg_{10}(\frac{N}{n_i})$$

Donde:

. N: Representa la cantidad total de documentos.

. n_i: Representa la cantidad de documentos donde ocurre el término i.

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

2.2 Búsqueda:

Al introducir la búsqueda (query) deseada y hacer click en el botón "Buscar", en caso de ser una búsqueda **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**. Luego se ubicará cada palabra normalizada de la query en el diccionario **QueryWeight** mediante el método **FeedQueryWeight** (ambos pertenecientes a la clase **Search**), donde a cada palabra se le asociará su peso. El cálculo del peso de cada término de la query está dado por la siguiente fórmula:

$$W_{ij} = (\alpha + (1-\alpha)\frac{freq_{iq}}{max_l freq_{lq}})\lg_{10}(\frac{N}{n_i})$$

Donde:

. W_{ij}: Representa el peso del término i en la query.

. $freq_{iq}$: Representa la frecuencia del término i en la query.

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

Luego se procede a buscar entre los documentos, aquellos que satisfagan total, o al menos parcialmente la búsqueda. Para ello se emplea la fórmula de Similitud de Cosenos:

$$Rsim(d_j, q) = \frac{\sum_{i=1}^{n} W_{ij} W_{iq}}{\sqrt{\sum_{i=1}^{n}(W_{ij})^2}\sqrt{\sum_{i=1}^{n}(W_{iq})^2}}$$

Donde:

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

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

. W_{iq}: Representa el peso del término i en la query.

3 Operadores

Si en alguna palabra de nuestra búsqueda apareciera algún/unos de los operadores [$\sim$! $\wedge$ *], estos influirán directamente en el puntaje final de los documentos. Haciendo uso del método **Clean** de la clase **Operators** se reconocerán dichos operadores y se procederá de la siguiente forma en caso de existir algún/unos de ellos:

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

- $\wedge$ Operador de **Solo 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 utiliza el método **Charcounter** de la clase **Operators**.

- $\sim$ 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 determinará la distancia entre ambas palabras mediante el método **Distance** de la clase **Operators**. De ser esta menor o igual a 10 palabras, su puntaje será considerablemente alto, de no ser así, se tendrá en cuenta al mostrar los resultados finales.

4 Resultados

Una vez ubicados los 3 documentos de mayor puntaje, estos serán mostrados en pantalla en orden de relevancia descendente, precedidos con una línea de texto donde aparezca la palabra introducida en nuestra búsqueda. Para ello se utilizó el método **Show** de la clase **Snippet**.
En caso de realizar una búsqueda donde las coincidencias sean nulas, se ofrecerá en pantalla una sugerencia de búsqueda, la cual será la palabra más parecida a la primera palabra de nuestra búsqueda. Para ello se utilizaron también los métodos de la clase **Snippet**:

1- **EditDistance**: Función basada en el algoritmo conocido como "Distancia de Levenshtein", el cual determina la mínima cantidad de ediciones a realizar para convertir una cadena de caracteres en otra.

2- **Suggestion**: Función que determina en base a los resultados obtenidos al aplicar el método **EditDistance**, la palabra con mayor similitud a la primera palabra de nuestra búsqueda.

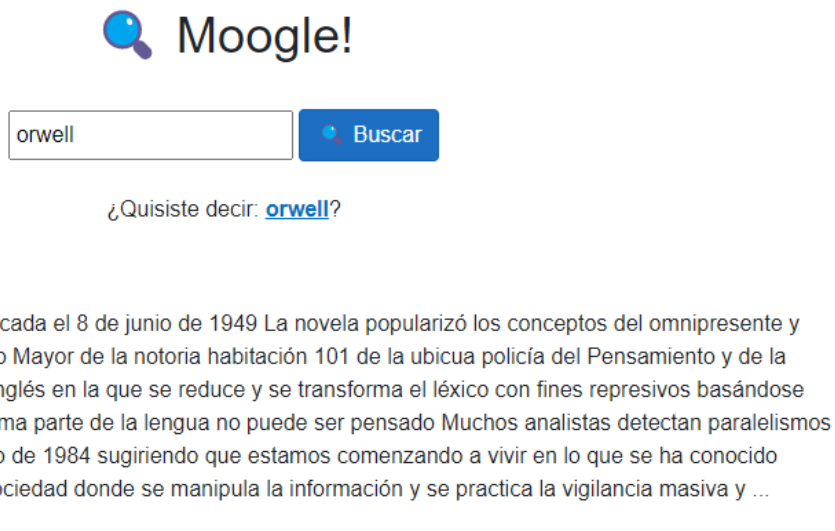
Manual de Usuario:

Al iniciar Moogle! y acceder a la página se nos muestra como en la siguiente imagen:

¿Cómo realizar una búsqueda con Moogle!?:

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

2- Hacer click en el botón Buscar.



En caso de no encontrar ningún resultado para nuestra búsqueda, podemos hacer click en la sugerencia que muestra la aplicación y así obtendremos un documento que podría estar relacionado con nuestra búsqueda.