Proyecto de Programación I. Moogle!

Facultad de Matemática y Computación.
Universidad de La Habana. Curso 2022.

Nombre: Marian S. Álvarez Suri.
Grupo: 121.

El Proyecto Moogle! consiste en un sistema de recuperación de información centrado en la búsqueda de cadenas de texto dentro de un conjunto de archivos de tipo Texto Plano.
La idea central del proyecto ha sido facilitar al usuario a la hora de realizar la búsqueda. Es por ello que cuenta con varias implementaciones que permiten ser más específicos en lo que se desea encontrar, mostrando una amplia gama de resultados y de alternativas.
La interfaz gráfica ha sido programada en su mayoría con HTML y CSS, mientras que el código de la búsqueda fue implementado en C#.

Haciendo un recorrido por el código del proyecto, encontramos las clases **Start**, **SearchQuery**, **Results**, **Moogle**, **SearchItem** y **SearchResult**.

La clase **Start** es la encargada de almacenar todos los archivos de texto plano (.txt) que se encuentran ubicados en la carpeta Content (y en todas sus subcarpetas) en una Lista que, a su vez, tendrá un listado de las palabras que contenga cada documento, o sea, una Lista de Listas. Este proceso será realizado por el método TxtReader de la propia clase **Start**.
Posteriormente, se procede a emplear el algoritmo *TF-IDF* (Term Frequency - Inverse Document Frequency) para determinar la relevancia que tiene cada palabra en un documento. Consiste en una medida que determina la frecuencia con que aparece una palabra en un documento y la relevancia que tiene en el vocabulario general. Así, mientras más frecuente sea una palabra en un documento, y menos frecuente en el vocabulario general, mayor será su *TF-IDF*.
En el proyecto se emplea un modelo vectorial para determinar qué tan relevante es un documento en comparación con la búsqueda realizada. Es por ello que la clase **Start** también calcula la magnitud de los documentos, elemento que será utilizado más adelante.
Estos datos son obtenidos antes de realizar la búsqueda y se almacenan durante todo el tiempo que se encuentre el proyecto en ejecución, permitiendo así que el tiempo de espera sea mínimo.

A continuación se procede a realizar un trabajo bastante similar con la búsqueda introducida por el usuario, auxiliándose de la clase **SearchQuery.** Una vez obtenida la magnitud del query, se procede a realizar el algoritmo de similitud de cosenos. La similitud de cosenos es una medida de la similitud existente entre dos vectores en un espacio con el que se evalúa el valor del coseno del ángulo comprendido entre ellos. Mientras más cercano a 1 se encuentre este valor, mayor será la importancia del documento en cuestión.
Una de las implementaciones realizadas en esta clase es el trabajo con los operadores de búsqueda que pueda introducir el usuario, los cuales afectan de manera significativa los resultados que se obtendrán, haciendo variaciones convenientes en la similitud de cosenos de los documentos.

En este punto se tendrán identificados los documentos más relevantes a la búsqueda, donde entra en juego la clase **Results** para obtener un fragmento del documento que contiene parte de la búsqueda efectuada.
Adicionalmente, si no se encuentran resultados luego de este proceso, el programa pasará a encontrar sinónimos de las palabras introducidas en la búsqueda original, efectuando todo el proceso nuevamente.
Si aún así no han sido encontrados documentos relevantes, se utiliza un algoritmo llamado Distancia Hammil que compara la búsqueda realizada con las palabras que se encuentran en los

documentos, buscando así cuál es la palabra que más se asocia a la búsqueda hecha y otorgando la opción al usuario de realizar una nueva búsqueda que sí tendrá resultados garantizados.