Proyecto de Programación Moogle!

Facultad de Matemática y Ciencias de la Computación.

Universidad de La Habana. Curso 2022.

Raciel A. Simón Domenech C-121

En este proyecto realizamos una aplicación cuyo objetivo es buscar inteligentemente un texto en un conjunto de documentos utilizando la tecnología .NET Core 6.0, Blazor como framework web para la interfaz gráfica y en el lenguaje de programación C#.

La búsqueda se realiza utilizando la Frecuencia de Término - Frecuencia Inversa de Documento (TF-IDF), primero se divide la búsqueda introducida por el usuario utilizando el método Score.QuerySpliter() de donde se obtiene un arrays de strings de las palabras de la búsqueda y otro que incluye caracteres especiales. En caso de que el usuario introduzca el carácter ‘~’ ,utilizando el método Score.IndexGetter() se guarda un array que contiene los índices de las palabras adyacentes a dicho carácter.

Posteriormente se utiliza el método Documents.FirstDocumentsGetter(); para inicializar un array de objetos del tipo Documents, los cuales contienen el título de los documentos, el contenido dividido en un array de strings y la relevancia.

Utilizando dicho array de documentos se procede a crear, con el método Score.GetTFsArr(); un array de objetos del tipo Words, los cuales contienen a cada palabra de la búsqueda, un array de doubles donde se guarda la Frecuencia de Término de cada palabra en cada documento, así como el double Frecuencia Inversa de Documento de cada palabra de la búsqueda.

En caso de ser necesario(si el usuario introdujo el carácter ‘~’ se procede a determinar, con el método Documents.ProximitySetter(); la proximidad de las palabras adyacentes a dicho carácter.

Posteriormente, con el método SpecialsSetter(); se le asigna a cada objeto del tipo Words su carácter especial correspondiente(si el usuario introdujo alguno).

Utilizando el método Documents.TotalScoreSetter(); se procede a calcular el Score de cada documento con el TF-IDF de cada palabra, posteriormente se invoca el método Documents.FinalScoreSetter(); el cual modifica el score según las condiciones que haya establecido el usuario(caracteres especiales)

Entonces se organizan los objetos de tipo Documents, empleando el método Documents.DocumentsSort(); el cual los organiza en orden descendiente según su Score.

Finalmente, utilizando el método Sugestion.Comparer(); se determina un array de strings semejantes a la búsqueda del usuario, y se devuelve un array de máximo cinco SearchItems ordenados según relevancia. Si la búsqueda no resultó satisfactoria, se procede a realizarla nuevamente empleando la sugerencia.