Moogle

Dayron Valdivia

Julio, 2023

Abstract

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' es una biblioteca de clases donde está implementada la lógica del algoritmo de búsqueda. Este proyecto tiene como objetivo buscar una consulta ingresada por el usuario en un conjunto de archivos de texto (con extensión '.txt') que estén en la carpeta 'Content'.

1 Descripción del Proyecto

El siguiente código a explicar es parte de una clase llamada "Moogle" que contiene un método estático llamado "Query". Este método toma una cadena de consulta como parámetro y devuelve un objeto SearchResult.

1.1 Diccionarios y variables

El código utiliza varios diccionarios y listas para realizar la búsqueda:

- La variable "NombreDeDocumentos" es un arreglo de cadenas que contiene los nombres de los documentos en la carpeta "Content".
- La variable "DireccionDeDocumentos" es un arreglo de cadenas que contiene las rutas completas de los documentos en la carpeta "Content".
- La lista "TodasPalabrasDeLosDocumentosSinRepetir" contiene todas las palabras únicas encontradas en los documentos en la carpeta "Content".
- Las listas "Debe", "NoDebe", y "Cerca" contienen las palabras clave especificadas en la consulta del usuario.
- La variable "Sugerencia" contiene una sugerencia de corrección ortográfica si la consulta del usuario contiene una palabra mal escrita.
- Las listas "CercaDos" y "snippet" se utilizan para generar fragmentos de texto relevantes para la consulta del usuario.

- El diccionario "Importancia" se utiliza para asignar una importancia a cada documento en función de su relevancia para la consulta del usuario.
- El diccionario "Palabra" se utiliza para contar el número de veces que aparece cada palabra en los documentos.
- El diccionario "ScoreDeDocumentos" se utiliza para almacenar el puntaje de relevancia de cada documento en función de su contenido y la consulta del usuario.
- El diccionario "PalabraQueMasSeRepitePorDocumentos" se utiliza para almacenar el número de veces que aparece la palabra que más se repite en cada documento.
- El diccionario "CantidadDeDocumentosQueApareceUnaPalabra" se utiliza para almacenar el número de documentos en los que aparece cada palabra.
- El diccionario "TFxIDF" se utiliza para almacenar el puntaje TF-IDF de cada palabra en cada documento.
- El diccionario "Vocabulary" se utiliza para almacenar la posición de cada palabra en cada documento.
- La variable "scoreOrdenado" se utiliza para almacenar los documentos ordenados por su puntaje de relevancia.

1.2 Método Rellenar

Posterior a las creación de las variables se implemento un método Rellenar que se encarga de como su nombre lo indica de completar estas listas y diccionarios. Se separa por partes para su mejor comprensión:

- Rellenar "Vocabulary" Este código recorre una lista de directorios que contienen documentos y crea un diccionario de vocabulario para cada documento. El diccionario contiene palabras como claves y una lista de las posiciones en las que aparecen en el documento como valores. Primero, se crea un diccionario vacío para cada documento. Luego, se lee el contenido del archivo y se divide en palabras utilizando el método Split(). Cada palabra se limpia utilizando el método LimpiarPalabra() posteriormente explicado. Luego, se comprueba si la palabra ya está en el diccionario. Si es así, se agrega la posición actual a la lista de posiciones. Si no, se agrega una nueva entrada al diccionario con la palabra como clave y una lista que contiene la posición actual como valor. Finalmente, el diccionario se agrega al diccionario Vocabulary.
- Rellenar "nombre y dirección de documentos" Este código crea un arreglo de strings llamado "NombreDeDocumentos" que contiene los nombres de

los documentos presentes en una lista de directorios. Para crear los nombres de los documentos, se utiliza la variable "DireccionDeDocumentos" que contiene la ruta completa del directorio en el que se encuentran los documentos. Se utiliza el método "Length" para obtener la longitud de la cadena "Content" que es la ruta relativa del directorio donde se encuentran los documentos. Luego se utiliza el operador ".." para retroceder dos niveles en la estructura de carpetas y obtener el nombre del documento. El resultado es un arreglo de strings que contiene únicamente los nombres de los documentos presentes en los directorios, sin la ruta completa.

- Rellenar todas las palabras de los documentos en "TodasPalabrasDeLosDocumentosSinRepetir" Este código recorre una lista de directorios (almacenados en la variable "DireccionDeDocumentos") y para cada uno de ellos, lee el contenido del archivo y lo divide en palabras utilizando el método "Split". Luego, recorre cada una de las palabras obtenidas y las limpia utilizando la función "LimpiarPalabra". Finalmente, verifica si esa palabra ya se encuentra en la lista "TodasPalabrasDeLosDocumentosSinRepetir" y, en caso contrario, la agrega a la lista. El resultado es una lista de todas las palabras únicas presentes en los documentos.

- Rellenar "PalabraQueMasSeRepitePorDocumentos" Este código recorre una lista de direcciones de documentos (almacenados en la variable "DireccionDeDocumentos") y para cada uno de ellos, inicializa un valor de 0 en la lista "PalabraQueMasSeRepitePorDocumentos". Luego, recorre cada una de las palabras obtenidas de la lista "Vocabulary[doc]" y verifica si la cantidad de veces que aparece esa palabra es mayor a la que está almacenada en "PalabraQueMasSeRepitePorDocumentos[doc]". Si es así, actualiza el valor almacenado en "PalabraQueMasSeRepitePorDocumentos[doc]" con la cantidad de veces que aparece esa palabra. El resultado es una lista que indica la palabra que más se repite en cada uno de los documentos.

- Rellenar "CantidadDeDocumentosQueApareceUnaPalabra" Este código recorre una lista de palabras obtenidas de todos los documentos sin repetir (almacenados en la variable "TodasPalabrasDeLosDocumentosSinRepetir") y para cada una de ellas, inicializa un valor de 0 en la lista "CantidadDeDocumentosQueApareceUnaPalabra". Luego, recorre cada uno de los documentos almacenados en "Vocabulary" y verifica si la palabra actual aparece en el documento. Si es así, incrementa un contador "a". Finalmente, almacena el valor de "a" en la lista "CantidadDeDocumentosQueApareceUnaPalabra" con la palabra actual como clave. El resultado es una lista que indica la cantidad de documentos en los que aparece cada una de las palabras obtenidas de los documentos sin repetir.

- Rellenar Diccionario "TFxIDFI" El código recorre una lista de documentos almacenados en la variable "DireccionDeDocumentos". Para cada documento, se crea un diccionario vacío llamado "TFxIDFI". Luego, para

cada palabra en la lista "TodasPalabrasDeLosDocumentosSinRepetir", se verifica si la palabra aparece en el documento actual (almacenado en "Vocabulary[item]"). Si la palabra aparece, se llama a la función "HallarTFxIDF" explicada posteriormente para calcular su valor TFxIDF y se agrega al diccionario "TFxIDFI" con la palabra como clave. Si la palabra no aparece, se agrega al diccionario con un valor de 0. Finalmente, el diccionario "TFxIDFI" se agrega al diccionario "TFxIDF" con el nombre del documento actual como clave. El resultado final es un diccionario que contiene los valores TFxIDF para cada palabra en cada documento.

- Rellenar "ScoreDeDocumentos" El código recorre una lista de documentos almacenados en la variable "DireccionDeDocumentos" y calcula un puntaje para cada uno de ellos basado en su similitud con una consulta dada. Primero, se verifica si el documento contiene todas las palabras obligatorias de la consulta y ninguna de las palabras que no deben estar presentes. Si el documento no cumple con estos criterios, se le asigna un puntaje de 0 y se pasa al siguiente documento. Si el documento cumple con los criterios anteriores, se verifica si contiene alguna palabra importante de la consulta. Si es así, se calcula su puntaje usando la función "Score" explicada posteriormente y se multiplica por un factor de importancia específico para esa palabra. Si el documento no contiene ninguna palabra importante, se verifica si contiene una combinación específica de dos palabras (almacenadas en la variable "elmtos") y si es así, se calcula un puntaje basado en la distancia entre las apariciones de esas dos palabras en el documento. Finalmente, se ordenan los documentos por su puntaje y se crea un diccionario "scoreOrdenado" que contiene los documentos ordenados de mayor a menor puntaje.

1.3 Otros Métodos

Ahora hablaremos sobre la implementación de otros métodos:

- "Snippet" El código recorre la lista de documentos ordenados por su puntaje y crea un fragmento de texto para cada uno de ellos que contiene las palabras importantes de la consulta y algunas palabras adicionales alrededor de esas palabras. Primero, se recorre la lista de palabras importantes de la consulta ("Palabra") y se verifica si el documento contiene cada una de ellas. Si es así, se crea un fragmento de texto que contiene la palabra y algunas palabras adicionales alrededor de ella (hasta 6 palabras a cada lado). Si el documento no contiene alguna palabra importante, se omite esa palabra en el fragmento de texto. Finalmente, se agrega el fragmento de texto para cada palabra importante al fragmento final del documento ("finalSnippet") y se agrega "..." si quedan más palabras importantes por agregar. Este proceso se repite para cada documento en la lista ordenada y se devuelve una lista de fragmentos de texto ("snippets") correspondientes a cada documento.

- "Score" Este código es una función que calcula el puntaje de relevancia de un documento en base a una consulta. Utiliza el método TF-IDF para calcular la importancia de cada palabra en el documento y en la consulta, y luego utiliza la fórmula de similitud del coseno para calcular el puntaje de relevancia. Primero, se divide el texto completo en palabras individuales y se limpian para eliminar cualquier carácter no deseado. Luego, se recorre una lista de todas las palabras en los documentos sin repetir y se calcula el puntaje TF-IDF para cada palabra en el documento y en la consulta. Finalmente, se utiliza la fórmula de similitud del coseno para calcular el puntaje de relevancia del documento en base a la importancia de las palabras en la consulta y en el documento. El resultado es un número entre 0 y 1, donde 1 indica que el documento es completamente relevante para la consulta y 0 indica que no es relevante en absoluto.

- "LimpiarPalabra" Este texto es una función que recibe una palabra y la limpia eliminando cualquier carácter no deseado al principio o al final de la palabra. Utiliza la función char.IsLetterOrDigit para verificar si un carácter es una letra o un dígito, y luego recorta la palabra para eliminar los caracteres no deseados. La función devuelve la palabra limpia.

- "HallarTFxIDF" Este código es una función que calcula el valor de la fórmula TFxIDF para una palabra en un documento específico. La función recibe dos parámetros: la palabra a calcular y el documento en el que se encuentra. Primero, la función verifica si la palabra existe en la colección CantidadDeDocumentosQueApareceUnaPalabra. Si la palabra existe, entonces se procede a calcular los valores necesarios para la fórmula. Los valores necesarios son: - valor1: la cantidad de veces que la palabra aparece en el documento específico. - valor2: la cantidad de veces que la palabra que más se repite aparece en el documento específico. - valor3: la cantidad total de documentos en la colección. - valor4: la cantidad de documentos en los que aparece la palabra. Una vez que se tienen estos valores, se procede a calcular el valor de la fórmula TFxIDF y se devuelve como resultado. Si la palabra no existe en la colección, entonces se devuelve 0.

- "HallarTFxIDFDelQuery" Este código es una función que recibe un texto y calcula el valor de la fórmula TFxIDF para cada palabra en el texto, en relación a una colección de documentos. Primero, se crea una lista llamada "textoEntero" que contiene todas las palabras del texto sin caracteres vacíos ni especiales. Luego, se verifica si cada palabra en "textoEntero" existe en un diccionario llamado "Palabra". Si la palabra no existe en "Palabra", se agrega con un valor inicial de 1. Si la palabra ya existe en "Palabra", se incrementa su valor en 1. A continuación, se itera a través de cada palabra en una colección de todas las palabras de los documentos sin repetir ("TodasPalabrasDeLosDocumentosSinRepetir"). Para cada palabra, se verifica si existe en "Palabra". Si la palabra existe, se calculan los valores necesarios para la fórmula TFxIDF (valor1, valor2, valor3 y valor4) y se calcula el resultado de la fórmula. Si la palabra no

existe, se asigna un valor de 0 al resultado. Finalmente, se devuelve un arreglo con los resultados de la fórmula TFxIDF para cada palabra en la colección de palabras de los documentos sin repetir.

- "sugerencia" Esta función recibe un texto y devuelve una sugerencia de corrección para cada palabra en el texto que no existe en una colección de todas las palabras de los documentos sin repetir ("TodasPalabrasDeLosDocumentosSinRepetir"). Primero, se itera a través de cada palabra en un diccionario llamado "Palabra". Si la palabra existe en la colección de palabras de los documentos sin repetir, se agrega al resultado. Si la palabra no existe en la colección, se busca la palabra más similar en la colección utilizando la distancia de Levenshtein (función "distanciaL" explicada posteriormente) y se agrega la palabra más similar al resultado. La distancia de Levenshtein es una medida de la diferencia entre dos cadenas de caracteres, es decir, el número mínimo de operaciones necesarias para transformar una cadena en otra. En este caso, se utiliza para encontrar la palabra más similar en la colección de palabras de los documentos sin repetir. Finalmente, se devuelve el resultado con las sugerencias de corrección para cada palabra en el texto que no existe en la colección de palabras de los documentos sin repetir

- "distanciaL" La función "distanciaL" implementa el algoritmo de distancia de Levenshtein. Toma dos cadenas de caracteres como entrada y devuelve el número mínimo de operaciones necesarias para transformar una cadena en otra.

- "busca1" La función "busca1" verifica si el primer carácter de una cadena está en un conjunto de operadores dados. Devuelve true si lo encuentra y false si no lo encuentra.

- "busca2" La función "busca2" itera a través de cada carácter en una cadena llamada "palabra". Si encuentra el carácter " ", devuelve true. Si no lo encuentra, devuelve false.

- "Operadores" Este código es una función llamada "Operadores" que recibe como parámetro una cadena de texto llamada "TextoCompleto". La función divide la cadena en palabras utilizando el carácter de espacio en blanco como separador y las almacena en un arreglo llamado "frase". Luego, la función itera a través de cada palabra en el arreglo "frase" y verifica si contiene alguno de los operadores especificados en el arreglo "operadores" utilizando la función "busca1". Si encuentra un operador, la función extrae el resto de la palabra a partir del índice donde se encuentra el operador y lo almacena en una lista correspondiente (Debe, NoDebe o Importancia) según el tipo de operador encontrado. Además, la función también utiliza la función "busca2" para verificar si la palabra contiene el carácter " ". Si lo encuentra, la función divide la palabra en dos elementos utilizando el carácter " " como separador y los almacena en un arreglo llamado "elmtos". Después de procesar todas las palabras

en el arreglo "frase", la función llama a otras funciones (Rellenar, Snippet y sugerencia) y asigna los resultados a variables correspondientes (snippet y Sugerencia). Finalmente, la función itera a través de un diccionario llamado "Palabra" y verifica si todas las palabras en el diccionario están presentes en una lista llamada "TodasPalabrasDeLosDocumentosSinRepetir". Si alguna palabra no está presente, la función llama a la función "sugerencia" para sugerir una corrección ortográfica para esa palabra y asigna el resultado a la variable "Sugerencia". Por último, la función crea un arreglo de objetos "SearchItem" utilizando los resultados de la función "scoreOrdenado", y los devuelve como un objeto SearchResult junto con la variable "Sugerencia".

2 Propiedades y limitaciones del proyecto

- El usuario puede buscar no solo una palabra sino en general una frase cualquiera.
- La consulta del usuario puede contener símbolos que no sean solo letras y números.
- Se pueden introducir operadores en las consultas, tales como:
- Se dará sugerencia al usuario a cerca de la consulta.
- Está diseñado para trabajar con un vocabulario en español.
- Para mayor eficiencia se sugiere que los archivos .txt no sean de gran extensión.