INFORME SOBRE EL PRIMER PROYECTO DE PROGRAMACION

MOOGLE

LEONARDO OJEDA HECHEVERRIA

C113

Lo probé con una base de datos de 25 mb de txt y se demoraba en cargar Tiene snippet ,puede buscar presionando la tecla “Enter” no tiene que hacer click en el botón de búsqueda. Pulsando la flecha de abajo podrá ver la búsqueda anterior que se hizo.

Cree un grupo de métodos para realizar la búsqueda.

1. GetDocuments(): Este método obtiene una lista de todos los documentos en una carpeta específica. Para hacer esto, toma la ruta de la carpeta y un patrón de archivo (en este caso, archivos de texto) y utiliza la clase Directory de C# para buscar y devolver una matriz de rutas de archivo.

2. GetTerms(string documentPath, bool document): Este método toma una ruta de archivo y un booleano que indica si el parámetro documentPath es una ruta de archivo o una cadena de texto. Luego, lee el archivo (si es una ruta de archivo) o toma la cadena de texto y realiza una serie de operaciones para obtener una lista de términos. Primero, elimina los caracteres de puntuación y las palabras vacías del texto usando las funciones Split y Where de C#. Luego, aplica una serie de operaciones de limpieza de texto como convertir todo a minúsculas y eliminar ciertos caracteres a cada palabra en la lista para obtener una lista de términos. Finalmente, devuelve la lista de términos.

3. GetTermFrequency(string term, string document, Dictionary<string, Dictionary<string, int>> invertedIndex): Este método toma un término, un documento y un índice invertido (un objeto Dictionary de Dictionary que mapea términos a documentos y sus frecuencias de aparición) y devuelve la frecuencia de aparición del término en el documento en el índice invertido. Para hacer esto, primero comprueba si el término y el documento están presentes en el índice invertido. Si es así, devuelve la frecuencia de aparición del término en el documento. De lo contrario, devuelve cero.

4. GetMatchedTerms(string document, List<string> queryTerms, Dictionary<string, Dictionary<string, int>> invertedIndex): Este método toma un documento, una lista de términos de consulta y un índice invertido y devuelve la cantidad de términos de la consulta que se ajustan al documento en el índice invertido. Para hacer esto, itera sobre cada término de la consulta y comprueba si la frecuencia de aparición del término en el documento en el índice invertido es

mayor que cero. Si es así, aumenta un contador de términos coincidentes. Finalmente, devuelve el número de términos coincidentes.

5. CreateInvertedIndex(string[] documents): Este método toma una matriz de rutas de archivo y crea un índice invertido que mapea términos a documentos y sus frecuencias de aparición. Para hacer esto, itera sobre cada documento en la matriz y llama al método GetTerms para obtener una lista de términos. Luego, itera sobre cada término en la lista y verifica si el término ya está presente en el índice invertido. Si no es así, lo agrega al índice invertido y crea un nuevo diccionario que mapea documentos a frecuencias de aparición para ese término. Si el término ya está presente en el índice invertido, verifica si el documento ya está presente en el diccionario de frecuencias de aparición para ese término. Si no es así, lo agrega al diccionario y establece su frecuencia de aparición en cero. Finalmente, aumenta la frecuencia de aparición del término en el documento en el índice invertido.

6. CalculateDocumentMagnitudes(Dictionary<string, Dictionary<string, int>> invertedIndex): Este método calcula la magnitud del vector para cada documento en el índice invertido. Para hacer esto, itera sobre cada documento en el índice invertido y para cada término en el índice invertido, agrega el cuadrado de la frecuencia de aparición del término en el documento al cuadrado de la magnitud del vector. Luego, toma la raíz cuadrada de la magnitud del vector y la almacena en un diccionario que mapea documentos a magnitudes de vector.

7. CreateQueryVector(string query, Dictionary<string, Dictionary<string, int>> invertedIndex): Este método crea un vector de consulta que mapea términos a sus pesos en base a su frecuencia y su idf. Para hacer esto, toma una cadena de consulta y el índice invertido.
8. CalculateScore, se utiliza para calcular el puntaje de un documento en función de un vector de consulta y un índice invertido. Toma cuatro argumentos: document es el nombre del documento para el que se está calculando el puntaje, queryVector es el vector de consulta que se ha creado a partir de la cadena de consulta, invertedIndex es el índice invertido que se ha creado para los documentos en la colección y documentMagnitudes es un diccionario que contiene las magnitudes de los vectores de cada documento.
9. CalculateQueryMagnitude, se utiliza para calcular la magnitud del vector de consulta. Toma un argumento queryVector, que es el vector de consulta que se ha creado a partir de la cadena de consulta. Primero, calcula la suma de los cuadrados de los pesos de los términos del vector de consulta y luego toma la raíz cuadrada de la suma para obtener la magnitud del vector de consulta.
10. GetSnippet, se utiliza para obtener un fragmento de texto de un documento que contiene la consulta. Toma dos argumentos: documentContent, que es el contenido del documento como una cadena, y query, que es la cadena de consulta. Primero, divide la cadena de consulta

en palabras individuales. Luego, busca la primera aparición de cada palabra en el contenido del documento y crea un fragmento de texto que contiene la palabra y las palabras circundantes. Devuelve el primer fragmento que se encuentra.

11. Exclued(string query)se utiliza para excluir documentos que contengan ! al principio de la palabra que se indica en la consulta. Toma un argumento query, que es la cadena de consulta.