Proyecto de primer semestre de Programación de la Facultad de Matemática y Computación de la Universidad de La Habana.

Ernesto Castellanos Vázquez

C – 113

Flujo de Datos:

La ejecución del programa comienza cuando desde el método Main de la clase program de MoogleServer se realiza una llamada a al método mainLoader de la clase Loader
esta clase posee tres propiedades estáticas ya que una vez establecido su valor no debe cambiar según la instancia de la clase, por la tanto se puede acceder a este incluso después de terminar la ejecución del método que se lo asignó:

matrix: una representación de las palabras en cada documento , cada documento se representa como un diccionario de pares palabra como key y una lista de objetos de tipo Position como value (que es solo un objeto usado para almacenar la posición de cada palabra).

documents: array que guarda la ruta y nombres de cada documento.

weight: guarda el peso de cada palabra en cada documento, utilizado para calcular la similitud cosénica.

El método mainLoader comienza cargando en la propiedad documents la ruta completa y nombre de los documentos luego mediante StreamReader guarda cada palabra de cada documento como una key en matrix y sus posiciones como un value luego recorre cada diccionario(documento) de matrix y cada palabra de esos diccionarios, para calcular el valor TFIDF de esa palabra en ese documento y almacenarlo en la propiedad weight como par palabra-peso.

Luego de esto el flujo vuelve al programa principal el cual realiza al realizar una búsqueda envía esta frase a Moogle.Query, este separa cada palabra de la frase guardando su posición y crea la sugerencia con las palabras que aparecen , en caso de que no aparezca una llama a Distance.minDist que devuelve la mas parecida de todos los documentos, el método también guarda en searchItem los nombres de los documentos(ya almacenados previamente), el snippet de la ultima palabra de la búsqueda presente en el documento llamando a Score.snippet, guarda el peso del documento en la consulta, multiplicando el valor devuelto por Score.score por el valor de los posibles operadores y ordena los resultados de la búsqueda según este peso, ademas de esto existen las siguientes clases:

La clase Operator que permite hacer el calculo de los operadores posee los métodos siguientes:

operatorValue: este método recibe gran cantidad de argumentos que son necesarios para implementar los operadores, llama a cada método según la existencia o no de operadores a la izquierda de la palabra, menos el de cercanía ya que este se comprueba a la derecha de la palabra para el operador de importancia solo devuelve la cantidad de asteriscos mas 1(ya que si solo aparece un asterisco retornaría 1 como valor de importancia).

#nota: en el caso del operador de cercanía solo se llama al método close si ambas palabras están presentes en caso contrario se devuelve 0 lo que provocaría que ese documento no sea devuelto en la consulta, esto ocurre de esta manera ya que se asume que si el usuario introduce este operador es porque desea que los documentos devueltos posean ambas palabras.

close: para el operador de cercanía utiliza un algoritmo similar a la parte merge del merge-sort para encontrar según las posiciones de las palabras el momento en el que estas estaban mas cerca entre si y devolver esa distancia.

isNot: simplemente devuelve true si la palabra no se encuentre en el documento, sirve tanto para el operador de existencia como para el no existencia.

La clase Score se usa principalmente para obtener el valor de los documentos en una búsqueda, contiene los métodos:

IDF: devuelve el IDF de la palabra solicitada para lo cual recorre cada uno de los documentos y comprueba si la palabra esta presente y devuelve el logaritmo del cociente entre la cantidad total de documentos y la cantidad total de documentos que contienen el término.

TFIDF: se vale del método anterior para calcular el tf-idf multiplicando la cantidad de elementos de la lista de posiciones(la cantidad de posiciones indica la cantidad de ocurrencias de esa palabra en el documento) por el idf del documento.

M: método sobrecargado que entrega la magnitud(valor utilizado para el calculo de la similitud cosénica) elevando cada termino al cuadrado o multiplicándolo por el termino correspondiente y sumando todos estos valores a medida que se encuentran.

score: devuelve el valor del documento para esa búsqueda valiéndose de la similitud cosénica calculada con la ayuda del método M.

snippet: recibe la ultima palabra de la búsqueda presente en los documentos y accede a la posición donde aparece la primera ocurrencia , posición que se encuentra en la lista de posiciones de la propiedad estática matrix de la clase Loader y mediante StreamReader lee el documento y guarda esa linea entera y la devuelve al método que la llamó.

La clase Distance tiene como función devolver la palabra mas cercana a la que se le solicita de todos los documentos para esto tiene dos métodos:

lev: este calcula la similitud entre dos palabras con un proceso recursivo que permite implementar el algoritmo levenshtein.

minDist: este recibe la palabra a buscar y mediante una serie de ciclos anidados compara esa palabra con cada una de las almacenadas gracias al método anterior y devuelve la mas parecida.

La clase ResultComparer hereda de IComparer y tiene como única función servir de criterio de comparación para ordenar los objetos de tipo SearchItem que debe devolver el método Query.