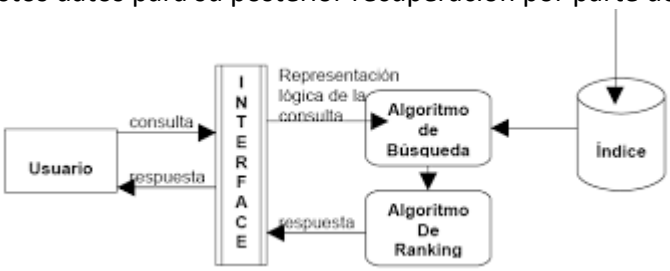
Proyecto de programación I.
Facultad de Matemática y Computación.
Universidad de La Habana. Curso 2023.

Javier Alejandro González Díaz
Grupo: C-122

La idea del uso de computadoras para la búsqueda de porciones relevantes de información en documentos se popularizo a raíz de un articulo “As We May Think” escrito por Vannevar Bush en 1945 que ha sido descrito como visionario e influyente , anticipando muchos aspectos de la sociedad de la información.En el articulo hablaba sobre un dispositivo denominado “Memex” que podría almacenar y recuperar información asociativa

Los Sistemas de Recuperación de Información (SRI) son un tipo de sistemas de información que tratan con bases de datos compuestas por diferentes tipos de objetos de información (Documentos textuales , artículos , imágenes , audios , videos , etc).Estos procesan las consultas de los usuarios permitiéndoles acceder a la información más relevante acorde a su búsqueda en un intervalo de tiempo apropiado. También se encargan del almacenamiento de estos datos para su posterior recuperación por parte de los usuarios.



En este proyecto no desarrollare a “Memex” pero si un (SRI) similar, no asociativo, sino basado en el modelo vectorial de recuperación de información el “Moogle!” .

En el proceso de recuperación de información se suelen distinguir las siguientes etapas:

1. Obtener representación de los documentos. Generalmente los documentos se presentan utilizando un conjunto más o menos grande de términos índice. La elección de dichos términos es el proceso más complicado.
2. Identificar la necesidad informativa del usuario. Se trata de obtener la representación de esa necesidad, y plasmarla formalmente en una consulta acorde con el sistema de recuperación.
3. Búsqueda de documentos que satisfagan la consulta. Consiste en comparar las representaciones de documentos y la representación de la necesidad informativa para seleccionar los documentos pertinentes.
4. Obtención de resultados y presentación al usuario.

Algoritmos de Búsqueda.

El Modelo Vectorial :

La búsqueda está basada en el modelo vectorial de recuperación de la información SRI, utilizando el TF-IDF (frecuencia de término - frecuencia inversa de documento), el cual determina la relevancia de una palabra asociada a un documento en una determinada colección, sumado a la Similitud del Coseno, método mediante el cual se asigna un peso a cada documento y se establece un ranking de resultados según la consulta del usuario.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

$$sim(i,j) = \cos\left(\vec{i},\vec{j}\right) = \frac{\vec{i} \bullet \vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

Fue definido por Salton [Salton, 1968] hace ya bastantes años, y es ampliamente usado en operaciones de recuperación de información. En el modelo vectorial se intenta recoger la relación de cada documento Di, de una colección de N documentos, con el conjunto de las m características de la colección. Formalmente un documento puede considerarse como un vector que expresa la relación del documento con cada una de esas características.

Para entender el funcionamiento hare una descripción sobre la estructura del programa :

Implementación MoogleEngine

Clases Principales

Clase Documento :

Esta clase es la encargada de gestionar todo lo relativo a los documentos sobre los que se realizaran las consultas. Al llamar a su constructor se le pasa el camino de uno de los documentos contenidos en la colección de la carpeta Content, el cual es cargado a la memoria y procesado. Al procesar el documento se eliminan las palabras vacías, se extraen y normalizan los términos y se guarda un listado con las raíces de estas.

Almacena el texto del documento, los términos que aparecen en el mismo. Brinda métodos para obtener una lista de las posiciones donde aparece un término en el documento, la cantidad de veces que aparece, si existe un término en el documento y la distancia mínima en el documento entre dos términos. También tiene métodos para obtener la cantidad de veces que aparece la raíz de la palabra y si existe en algun documento. El método ExtraerOracion extrae del texto del documento una oración con la mayor cantidad de palabras posibles de una lista que le pasamos (términos que
aparecen en la consulta) y que se utiliza para buscar el (snipet)fragmento del texto que se devuelve junto con los resultados de la búsqueda.

Clase QUERY :

Esta clase es la encargada de gestionar todo lo relativo a las consultas. Al llamar a su constructor se le pasa la consulta, de la que se eliminan las palabras vacías, se normalizan los términos y se procesan los operadores.
Almacena el texto de la consulta y los términos que aparecen en la misma.

Brinda métodos para obtener los términos que aparecen en la consulta y la cantidad de veces que aparece cada uno, los términos obligatorios (operador ^), los términos prohibidos (operador !), los términos importantes (operador *) y los términos cercanos (operador ~).

Clase Núcleo:

Esta clase es el núcleo del proyecto. Realiza el proceso de la búsqueda de la consulta en los documentos de la colección y calcula los pesos de cada uno.
Al llamar a su constructor se crea la lista de documentos, los cuales se procesan y quedan listos para las consultas. También se crea la lista de palabras vacías y se inicia el mecanismo para buscar los sinónimos. Este método es invocado una vez cuando se inicia la aplicación y deja ordenada toda la información para cuando se realicen las consultas.
El método RealizarConsulta es el utilizado para realizar una consulta al grupo de documentos de la colección. Se le pasa como parámetro la consulta que se va a realizar y retorna un arreglo de SearchItem con los resultados de la consulta realizada. En la propiedad Sugerencia de esta clase se devuelve una sugerencia al usuario en caso de que la búsqueda no coincida con los datos almacenados.

Al realizar el usuario una consulta, el proceso realizado por el programa es el siguiente:

1. Del total de documentos de la colección, buscamos los documentos que tengan al menos un término de la consulta. Si no tienen ningún término, el peso del documento será 0, por lo que no nos interesan ya que no aportan ninguna información.
2. Los documentos obtenidos en el paso anterior son filtrados para eliminar del grupo aquellos que tengan términos prohibidos o no tengan términos obligatorios (según los operadores ! o ^ que aparezcan en la consulta).
3. Se crea el vector TF de la consulta.
4. Se crea la matriz con los vectores TF de los documentos.

5. Realizamos la revisión de la matriz TF de los documentos antes de pasarla al cálculo de los pesos , debido a que si hay alguna columna en cero significa que el término no se encontró en ningún documento y esto haría una división por 0 que invalidaría el cálculo.

"La revisión se realiza en tres pasos que son los siguientes: "

Paso 1: buscamos las columnas que están en cero y si hay alguna buscamos si la palabra está mal escrita y nos da el sistema una sugerencia. Para buscar la sugerencia buscamos en todas las palabras de los documentos aquella con un porciento de similitud mayor con respecto a nuestra palabra. Si hay una sugerencia, se devuelve una sugerencia al usuario y calculamos la frecuencia del nuevo término en los documentos, actualizando la columna de la matriz TF.

Paso 2: buscamos las columnas que están en cero y si hay alguna buscamos la raiz de la palabra con el método STEMMING. Luego calculamos la frecuencia de esta en los documentos, actualizando la columna de la matriz TF. La cantidad de veces que aparece la raiz la multiplicamos por 2, para que sea menor el peso que del término original.

Paso 3: buscamos las columnas que están en cero y si hay alguna no queda más opción que eliminarla del vector TF-IDF de la consulta y de la matriz de TF-IDF de los documentos para esto se utiliza la clase Matrix que posee entre otros métodos , unos destinados para eliminar celdas de ciertas filas de una matriz y para remover varias columnas de una matriz.

6. Se realiza en cálculo de los pesos de los documentos utilizando el coseno del ángulo formado por el vector de la consulta y el vector de cada uno de los documentos.
7. Se ordenan los documentos de mayor a menor de acuerdo a los pesos de cada uno.
8. Se procede a crear el arreglo de SearchItem, incluyendo fragmento del texto que se devuelve junto con los resultados de la búsqueda.
9. Se devuelven los resultados.

Clases Auxiliares

Estas clases brindan funcionalidades al proyecto, encapsulando procesos necesarios para las clases principales, evitando repetir código en cada una de ellas.

Clase Matrix:

Esta clase contiene métodos como multiplicación matricial , suma , entre otros , para facilitar el calculo del score y garantizar un mejor entendimiento e implementación de este modelo vectorial. Ademas posee métodos que remueven las columnas vacias de nuestros calculaos para evitar invalidaciones de cálculos y por ende evitar fallas en el funcionamiento del programa en tiempo real.

Clase Herramientas :

Esta clase es la encargada de brindar al resto de las clases propiedades y métodos que son utilizados de forma indistinta. En ella se definen los caminos de los archivos de los documentos, métodos básicos del procesamiento de los textos y de cálculos, tipos de datos nuevos, Distancia de Damerau-Levenshtein etc.

Clase Stemming:

Esta clase es la encargada de sacar las raíces de las palabras . Es una variación del algoritmo de Porter para el español, publicada en jesusutrera.com/articles/article01.html. La misma se ha modificado para adaptarla a nuestro programa.
El algoritmo de Porter [Porter, 1980], se basa en una serie de reglas condición / acción. Tiene la finalidad de obtener la raíz de una palabra (lexema) y consta de una serie de pasos, en cada uno de los cuales se aplican sucesivamente una serie de reglas (reglas de paso) que van suprimiendo sufijos o prefijos de la palabra hasta que nos quedamos sólo con el lexema final.

Clase Stopwords:

Esta clase busca las palabras vacías en los documentos creando los valores de la propiedad Stopwords de la clase Herramientas . Brinda el método CreaListado() que devuelve como salida el listado de las palabras vacías que aparecen en los documentos de la colección.
Debido a que en la consulta pueden aparecer palabras vacías como lo artículos, las preposiciones, las conjunciones, etc., que no necesariamente aparecen todas en los documentos, comenzamos el proceso adicionando a nuestra lista los artículos, las preposiciones, las conjunciones, los pronombres y los adverbios del español.
Después creamos una lista de las palabras que aparecen en los documentos y de las veces que aparece cada una. Tomando en cuenta las veces que aparece cada palabra, ordenamos este listado de menor a mayor y buscamos la mayor cantidad posible de palabras que aparezcan en los documentos y que la suma de sus apariciones sea menor que la suma de las apariciones de las palabras que más aparecen. Esto nos brinda un punto de corte para diferenciar las palabras vacías de las palabras buenas. Las palabras que no estén dentro de las palabras buenas y que serán también las que más veces aparecen, las tomamos como palabras vacías.

Funcionalidades adicionales:

Operadores

El proyecto cuenta con varios operadores para mejorar la búsqueda del usuario:

· Exclusión, identificado con un ! delante de una palabra, indica que la palabra no debe aparecer en ningún documento devuelto.
· Inclusión, identificado con un ^ delante de una palabra, indica que la palabra debe aparecer en todos los documentos devueltos.
· Mayor Relevancia, identificado por varios * delante de una palabra, indica que la palabra es más relevante que las demás palabras de la consulta tantas veces como * tenga delante de ella.
· Cercanía identificado con un ~ entre las palabras indica que los documentos en los que aparecen estas dos palabras más cerca tendrán mayor peso.

Sugerencia

En caso de que la búsqueda no coincida con ningún valor almacenado , tenemos este "corrector " para garantizar de que aun habiendo palabras erróneas el programa intuya el termino correcto que el usuario quiso expresar en la consulta.No es intuición , esto es posible debido al método Distancia Damerau-Levenshtein ,que calcula la cantidad de modificaciones que hay que realizar en una palabra para llegar a otra de lo que podemos deducir que a menor distancia mayor similitud , y ahí esta la sugrencia.
Esta también se apoya en el Stemming para realizar una segunda búsqueda de sugerencia , ya que pueden existir casos como :
Caballo y caballería , el lexema de esta palabra es "caball" pero si hubiésemos intentado escribir "caballería" y la palabra relacionada con los documentos es "caballo" por dicho método Damerau-Levenshtein esa no seria la sugerencia mas exacta por lo que extraemos la raíz de la palabra y luego volvemos a aplicar el método ya que solo de "caball" a "caballo" hay una modificación y en este caso esa si seria la sugerencia.