Universidad de La Habana

Facultad de Matemática y Computación

Sistema de Recuperación de Información

Andy Sánchez Sierra C-511

Eduardo Moreira González C-511

Yadiel Felipe Medina C-511

Trabajo final de Sistema de Recuperación de Información

2021-2022

1. Interfaz de Usuario

La interfaz de usuario implementada, consta de una aplicación de escritorio desarrollada en Python usando la biblioteca PyQt5 contenida en el módulo application.py.

Al iniciar la aplicación se mostrará un diálogo para seleccionar el directorio raíz sobre el cuál serán realizadas las consultas. Una vez seleccionado dicho directorio, se mostrará la ventana principal de la aplicación (Fig. 1) la cual contiene el resto de las funcionalidades requeridas. La ventana de la figura 1, contiene un cuadro de texto para escribir las consultas.

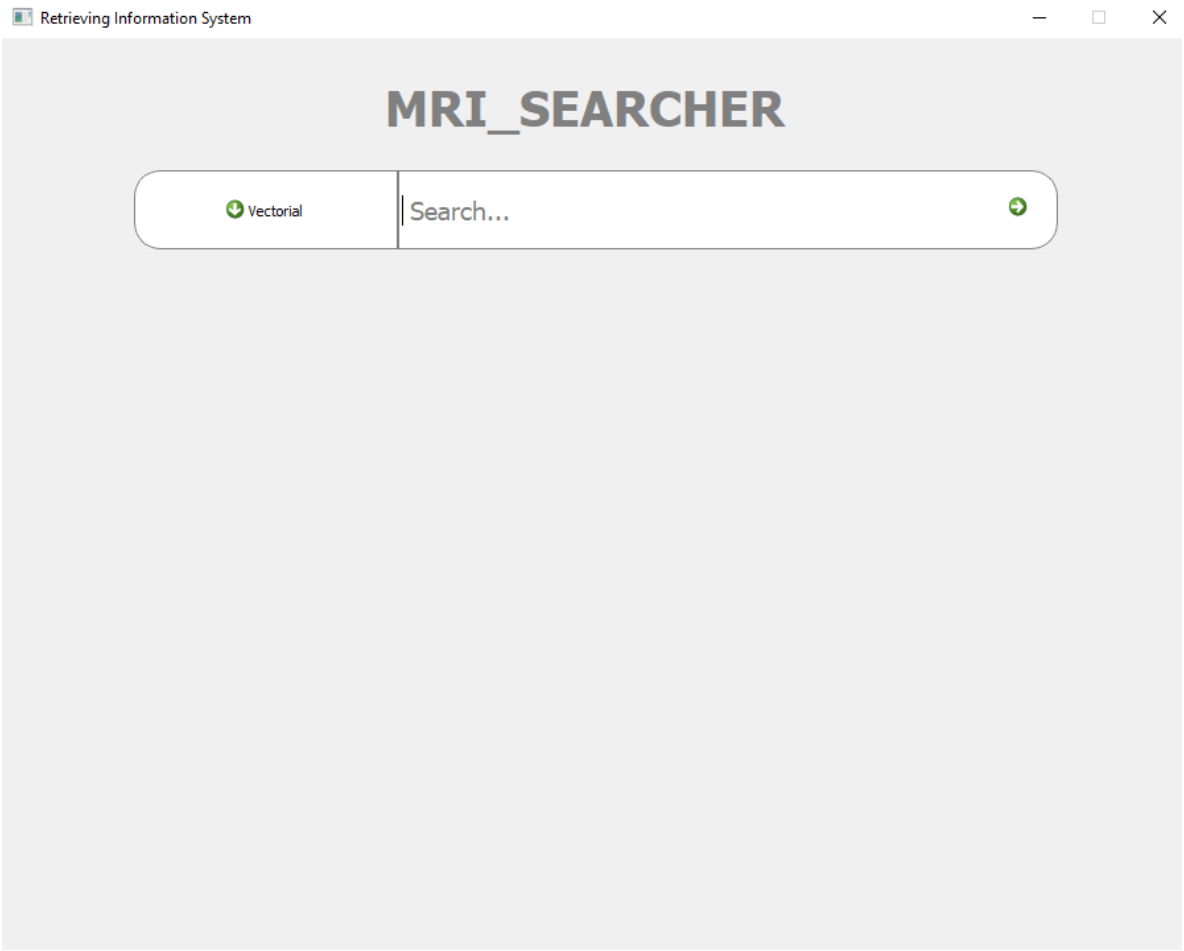

Figura 1: Ventana principal de de Interfaz de Usuario.

1.1. Consultas

Para realizar una consulta solo es necesario escribir el cuerpo de la misma en el cuadro de texto y presionar Enter, de esta forma la query será procesada y serán mostrados solo los documentos detectados como relevantes para la consulta en

cuestión (Fig. 2). Después de presionar Enter, se mostrará un lista con los nombres de los documentos recuperados. Para visualizar uno de los documentos mostrados solo es necesario hacer doble click sobre él.

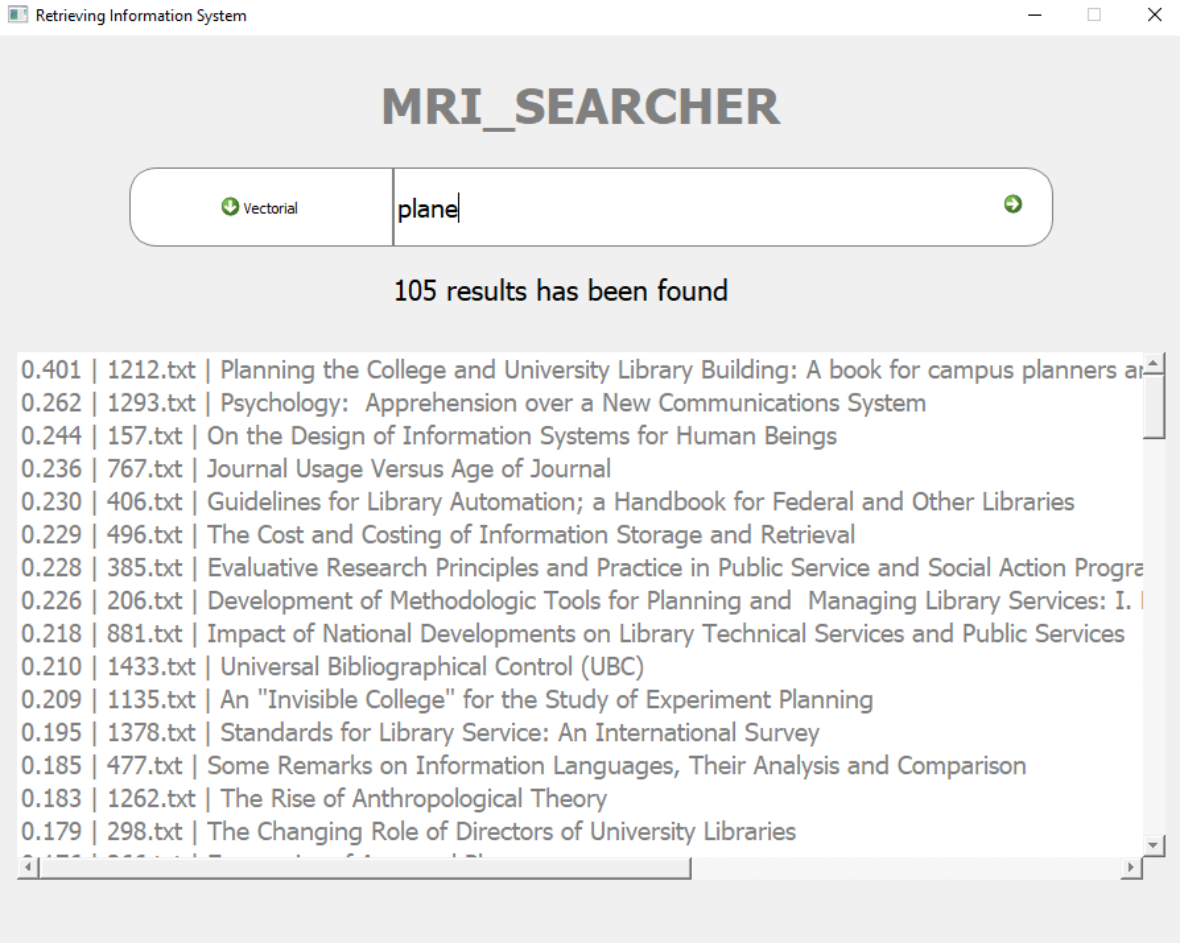

Figura 2: Resultados obtenidos por el sistema para una consulta de prueba.

1.2. Autocompletado

A la hora de realizar una consulta, la aplicación autocompleta por palabras del corpus, para facilitar la creacion de las consultas.

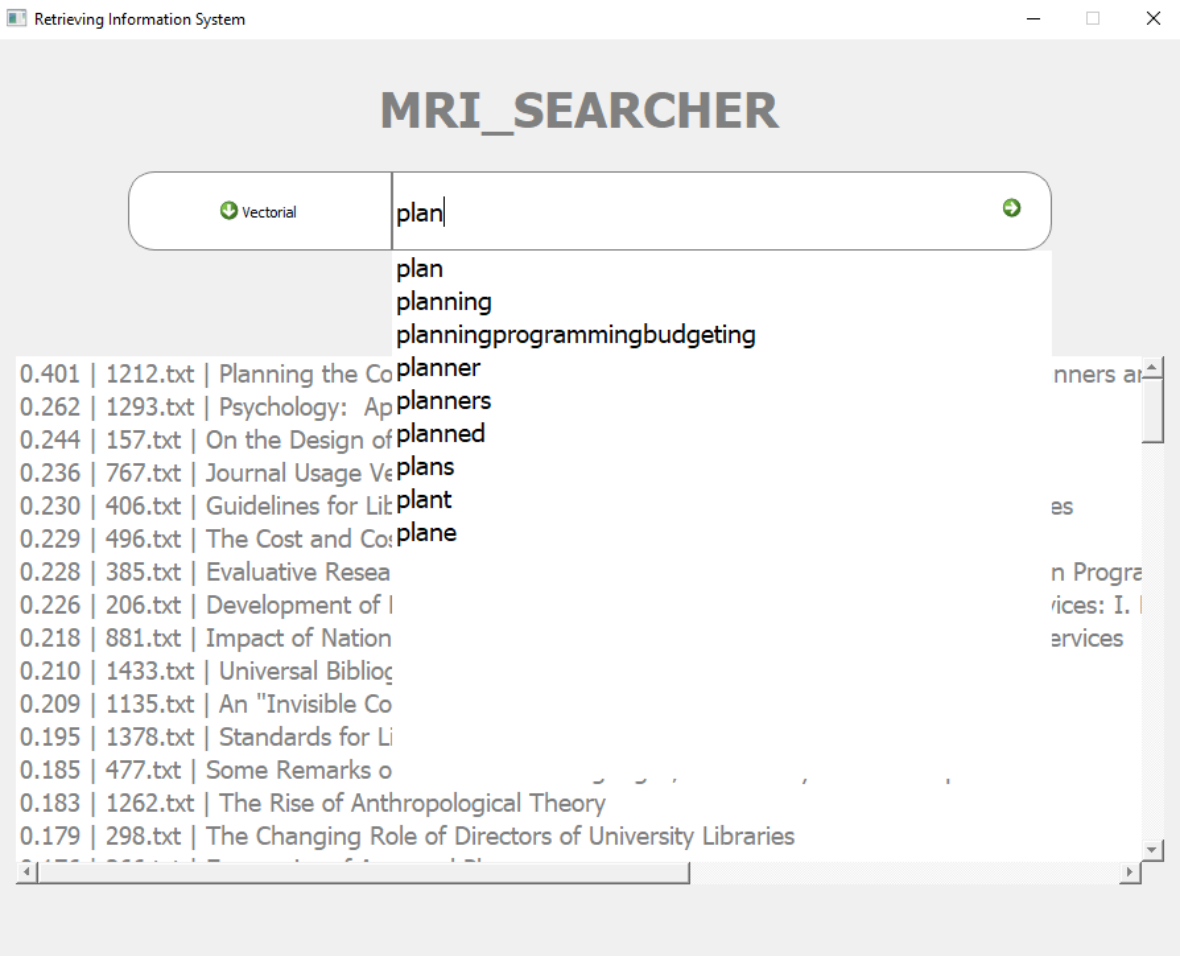

Figura 2: Resultados obtenidos por el autocompletado para una consulta de prueba.

2. Procesamiento de Texto

El procesamiento de los textos recibe como entrada una cadena de literales. Lo anterior significa que hay que convertir los documentos en formato .pdf de la colección a .txt utf-8. Para extraer el texto de un documento .pdf se usó la biblioteca pdfminer3 y es este texto el que recibe como entrada el módulo de procesamiento de texto. Este módulo usa la biblioteca nltk para procesar el texto. El procesamiento de un texto es un procedimiento que se puede dividir en las siguientes operaciones:

2.1. Análisis Léxico

Primeramente se separa el texto en oraciones y posteriormente estas se separan en palabras(en este paso las palabras unidas por guión se separan). Con el texto

representado como una lista de palabras se computa la parte de la oración que representa cada palabra en el texto dado, para poder realizar un análisis más preciso(correcto) en futuras operaciones como por ejemplo: la lematización de una palabra, la cual depende de la función gramatical de la misma dentro de la oración en la que aparece. Además se eliminan los números y los símbolos presentes en el texto, así como todas aquellas palabras que no desempeñen una funcionalidad semántica relevante dentro de la oración donde aparecen. Por ejemplo los sustantivos frecuentemente aportan mayor semántica que los adjetivos, adverbios y verbos.

2.2. Eliminación de stopwords

Esta operación se realiza con el objetivo de eliminar las palabras que aportan poca información al proceso de recuperación de información debido a su alta presencia en los documentos de la colección como por ejemplo: los artículos, pronombres, etc.

2.3. Lemmatization

Esta operación se usa para eliminar afijos(prefijos y sufijos) de una palabra y de esta forma permitir identificar palabras las cuales se diferencian en pequeñas variaciones sintácticas como la misma palabra. Este proceso de lematización computa la forma base de una palabra, la cual siempre es una palabra válida en el diccionario. Además para computar la forma base de una palabra es necesario conocer la parte de la oración que esta representa.

2.4. Stemming

Esta operación computa una forma base de una palabra que no tiene porqué ser necesariamente una palabra válida en el diccionario.

3. Modelo de Recuperación de Información

Se proponen dos modelos de recuperación de información, el vectorial generalizado, pero con una modificación para mejorar su eficiencia, fundamentalmente en memoria y el vectorial, por su posibilidad de ser usado por usuarios no expertos y por ser un modelo eficiente con repositorios grandes.

.

La principal dificultad en la implementación del modelo generalizado es que los documentos y los términos se representan como un vector de 2^t componentes, donde t es la cantidad de términos en la colección de documentos y como t suele ser un número grande puede que no alcance la memoria RAM para guardar todos los vectores. Para generar los vectores de los términos $\overrightarrow{k_i}$ se hace:

$$\overrightarrow{k_i} = \frac{\sum \forall r, g_i(m_r) = 1, C_{i,r}\overrightarrow{m_r}}{\sqrt{\sum \forall r, g_i(m_r) = 1, {C_{i,r}}^2}}$$

Donde:

$$C_{i,r} = \sum_{d_j | g_l(\overrightarrow{g_j}) = g_{j(m_r)}\ for\ all\ l} W_{i,j}$$

El vector $\overrightarrow{m_r}$ es un vector unitario de 2^t componentes, que le corresponde al $minterm\ m_r$ en el espacio de dimensión 2^t. Como a lo sumo hay N $minterm$ activos, donde N es la cantidad de documentos en el córpus, entonces se puede reducir la dimensión de este espacio a M, donde M es la cantidad de $minterm$ activos ($M \leq N$) y representar el vector m_r como un vector unitario de M componentes, para el cual la componente en 1 es la que le corresponde al $minterm\ m_r$ en la lista de $minterm$ activos.

3.1. Creación de Índices

Para la recuperación eficiente de los documentos se necesita crear índices que faciliten los cálculos.

Para el modelo vectorial, el método que se propone es el de Matriz de asociación, donde se guarda la relevancia de los términos en los documentos.

$$W_{i,j} = relevancia\ del\ término\ t_i\ en\ el\ documento\ d_j$$

Para el modelo vectorial generalizado, el modelo hace uso de varios diccionarios como: docs_to_term que para cada término guarda la lista de documentos en los que aparace, docVects que guarda para cada documento el vector de t componentes que lo representa y termVectors que guarda el vector de t componentes asociado a cada término. Además tambien se almacenan otros diccionarios auxiliares como index_of_term y docs_names. Todos los diccionarios se guardan en formato Json para facilitar su uso.

4. Evaluación de los Modelos

Para la evaluación del sistema se utilizó la colección cranfield con 1400 documentos y 225 consultas y la colección Cisi con 1460 documentos y 112 consultas.

Se implementaron las medidas de precisión, recobrado, f-medida, f1-medida y r-precisión. Para evaluar el modelo se ejecutaron 25 consultas y se promediaron los resultados de las mismas para cada medida implementada, arrojando los siguientes resultados:

Cranfield -> Vectorial:

1. Precisión: 0.0054
2. Recobrado: 0.6284
3. F-Medida: 0.0067
4. F1-Medida: 0.0107
5. R-Presición: 0.0279

Cranfield -> Vectorial Generalizado:

1. Precisión: 0.0161
2. Recobrado: 0.1067
3. F-Medida: 0.0184
4. F1-Medida: 0.242
5. R-Presición: 0.0120

5. Funcionalidades Opcionales

Como funcionalidad opcional en nuestro sistema se implementó la expansión de las consultas, Integración de algoritmos de Crawling y el uso de conocimientos como ontologías o tesauros, ademas de la realización de un Trie para el autocompletado a la hora de crear las consultas.

5.1. Expansión de consultas

La expansión de consultas se implementa combinando los análisis global y local. Para el análisis global se busca una lista de sinónimos por cada palabra en la

consulta en un tesauro y para el local se computa una matriz de correlación entre los términos(a través de los documentos en los que co-ocurren). La estrategia de expansión de consultas resultante de la combinación de las anteriores es la inclusión en la consulta para cada palabra de la misma de las k palabras con mayor valor de correlación, además mayor que un valor fijado(threshold), que se encuentran en la lista de sinónimos que brinda el tesauro.

5.2. Crawling

Para aumentar el dataset, se acude a la integración de algoritmos de Crawling, que dado una url, $name$ y k, crea una carpeta de nombre $name$ y dentro guarda k documentos y sus títulos, que son los devueltos por el algoritmo de Crawling.

5.3. Autocompletado

Para realizar el autocompletado de las palabras en la consulta, se realiza un Trie en el que se guardan todas las palabras del corpus.