**Tarea Programada #1**

**Estudiantes:**

**Marco Antonio Reveiz Rojas 2019053583**

**Sebastián López Herrera 2019053591**

**Instituto Tecnológico de Costa Rica**

**Escuela de Ingeniería en Computación**

**IC-8020** **Recuperación de Información Textual**

**2021**

**Introducción**

En el proyecto se debía implementar un sistema que fuera capaz de realizar consultas a una colección de archivos denominada “man.es”. Los resultados de las consultas se dan utilizando el modelo probabilístico llamado BM25, desarrollado en los años 70 por Spetehn E. Roberton y Karen S. Jones. Este proyecto consiste en la programación de tres herramientas que trabajarían juntas para procesar los documentos, realizar las consultas y explorar los resultados. Las indicaciones del proyecto recomendaban implementar las herramientas por medio de los lenguajes de programación Python o Pearl. Los documentos de la colección se almacenan recursivamente por lo que las herramientas del sistema dependen una de otra. Se debieron tener en cuenta una serie de restricciones para leer cada documento con el fin de que las consultas no se alteraran. El objetivo principal del sistema es que se puedan realizar consultas a todos los archivos de la colección general para demostrar la capacidad del modelo probabilístico BM25.

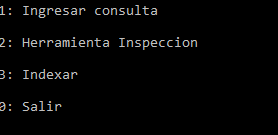
**Ambiente de Desarrollo**

El desarrollo del sistema desde un principio se acordó en implementar el lenguaje de programación Python, ya que al contar con tipos de datos como diccionarios se asumió que se podrían implementar archivos en formato de texto llamado JSON. Este formato facilitaría la forma en que se puede guardar la información de cada documento para leerlo a la hora de realizar el intercambio de datos en las consultas. En Python existe un paquete incorporado que sirve para manejar el contenido de un archivo con formato JSON, el cual implementa métodos que aceleran el proceso de lectura de información una vez ocurra la primera etapa del proyecto.

Esta primera etapa llamada indización, se decidió leer los archivos con una librería llamada “CSV” que por lo general es implementada para importar y exportar formatos de bases de datos. Este modulo implementa clases que se encargan de leer y escribir en archivos en este caso con terminaciones que van del .1 al .8. Una vez esta librería lee los archivos correctos, su información es transformada en diferentes clases de creadas de acuerdo con el tipo de información que requiere guardarse. Por ejemplo, se crearon estructuras para almacenar los datos de una colección que debe tener una lista de estructuras encargadas de almacenar los datos de los documentos. El sistema se basa que en su primera etapa los datos de colecciones y documentos se convierten en estructuras. Al finalizar esta etapa, la información de toda la colección general se almacena en tres archivos con formato JSON denominados colección con la información de todas las colecciones, un archivo documentos con la información de todos los documentos válidos, y el vocabulario que se forma a partir de una clase denominada de la misma manera que se va creando conforme se van leyendo los documentos y extrayendo los términos. Las clases creadas también cuentan con métodos que ayudan a calcular los datos necesarios para las fórmulas del modelo BM25 como el cálculo del valor de “idf”.

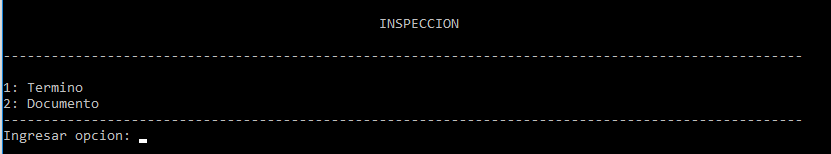
**Instrucciones para Ejecución**

Una vez se empieza a ejecutar el sistema se muestra un menú principal el cual tiene las siguientes opciones enumeradas:



La primera opción es para digitar la consulta que el usuario desea ingresar. Para realizar la consulta se necesitan los siguientes datos la ruta del directorio, la cantidad de documentos, el nombre del archivo a guardad y por ultimo la consulta.

La segunda opción se dedica a la herramienta de inspección que muestra los datos de un termino o un documento determinado. Para buscar uno en específico solo se digita el termino que desea buscar o el nombre del documento con su terminación.





La tercera se podría considerar como la más importante debido a que esta es la encargada de realizar la indización que se encarga de cargar los archivos a estructuras y realizar todo el proceso antes de que las consultas puedan realizarse.

**Consultas**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Consulta** | **Enunciados de relevancia** | |
| **Q1** | colas de mensajes para la comunicación entre procesos | ipc.2  ipc.5  msgctl.2  msgget.2  msgop.2  msgrcv.2  msgsnd.2  recv.2 | recvfrom.2  recvmsg.2  sendmsg.2  sendto.2  socket.2  socket.7  tcp.7 |
|  |  |  |  |
| **Q2** | segmentos de memoria compartida | shmat.2  shmctl.2  shmdt.2  shmget.2 | shmop.2  ipc.5  ipc.8 |
|  |  |  |  |
| **Q3** | @changes @verbose | chgrp.1  chmod.1  chown.1 |  |

Los archivos resultados fueron encontrados en los primeros 30 puestos del escalafón.

**Estado**

|  |  |  |  |
| --- | --- | --- | --- |
| **Funcionalidad** | | **SI / NO** | **Observaciones** |
| **Indización** | |  |  |
|  | Indiza todos los archivos de la colección | SI |
|  | Crea archivos en almacenamiento secundario | SI |
|  | Concatena las palabras separadas con guión especial. | SI |
|  | Convierte a minúsculas y quita tildes | SI |
|  | Preserva eñe | SI |
|  | Acepta palabras con dígitos | SI |
|  | Acepta números con puntos (1.2 2.3.4) | SI |
|  | Elimina stopwords | SI |
|  | Convierte \-\- en @ | SI |
|  | Guardará archivos generados en el directorio especificado | SI |
| **Búsqueda** | |  |  |
|  | Aplica mismas transformaciones a las consultas | SI |
|  | Usa el mismo archivo de stopwords para consultas | SI |
|  | Similitud BM25 implementada | SI |
|  | Genera escalafón completo, con la información pedida | SI |
|  | Genera página web, con la información pedida | SI |
| **Herramienta de inspección** | |  |  |
|  | Muestra la información almacenada para un documento dado | SI |
|  | Muestra la información almacenada para un término dado. | SI |

**Comentarios Finales**

El desarrollo del sistema fue bastante sencillo debido a que se organizaron los datos en clases creadas para manipular su información. Sin embargo, a la hora de ejecutar la etapa de indización su procedimiento era sumamente largo con una duración de al menos 40 minutos. Los programadores del sistema consideraron la idea de implemetar hilos a la hora de leer las colecciones recursivamente, pero no hubo tiempo para optimizar el proceso.

Otro aspecto que se tomo en cuenta es a la hora de restringir los archivos de cada colección. Debido a que si algunos de estos no cumplían con el formato requerido, se eliminan del todo y no se agregan a los archivos JSON por lo que no se revisan a la hora de realizar las consultas. Esto puede provocar que hayan algunos cambios en los resultados de las consultas realizadas.

Los archivos JSON son sumamente sencillos de utilizar a la hora de almacenar informacion y manejarla con el lenguaje de programación PYTHON debido a la existencia del tipo de datos diccionario. Este ayuda a almacenar la informacion de manera clara para volver a cargarla a memoria y realizar calculos o consultas.

En conclusion general, el programa quedo concluido, con todas las caracteristicas solicitadas en las especificaciones. La principal limitante fue el tiempo de ejecucion ya que la base de datos con los documentos era demasiado grande. Lo que nos obligo a optimizar la leida y creacion del indexado lo mas posible.