BÚSQUEDA Y MINERÍA DE INFORMACIÓN

PRÁCTICA 3

Búsqueda avanzada: proximal y PageRank

Tomas Higuera Viso
Miguel Antonio Núñez Valle
Pareja 25

Pregunta 1

- Hemos realizado la implementación del **motor de búsqueda proximal**, que realiza tanto consultas proximales como literales.
- Hemos implementado el **índice posicional**, así como todas las estructuras relativas al mismo, explicadas más adelante en la memoria.
- Hemos implementado el algoritmo **PageRank**, aunque no hemos conseguido obtener resultados, al hacer pruebas con el grafo procedente de Google.
- Hemos implementado un **crawler**, que al igual que con el índice posicional, esta explicado mas adelante en la memoria.

Pregunta 2

Para la creación del índice posicional, primero hemos tenido que implementar un diccionario nuevo para poder manejar las posiciones de los términos en documentos. Este nuevo diccionario se encuentra en:

- es.uam.eps.bmi.search.index.structure.impl.PositionalDictionary

Este nuevo diccionario es un mapa que guarda como clave un String, que es el término de la colección y como valor una PositionalPostingsList que contiene una lista de PositionalPostings del término.

En cuanto a PositionalPostingsList, hemos tenido que crear esta clase para poder almacenar una lista de PositionalPostings, así como, hemos tenido que crear un iterador propio para poder acceder a los elementos de la lista correctamente. Ambas clases se encuentran implementadas en el paquete:

- es.uam.eps.bmi.search.index.structure.positional.impl

Una vez implementadas estas clases hemos realizado el índice que se nos pide en este ejercicio. El constructor de este índice extiende la clase BaseIndexBuilder, para poder hacer uso de algunos de los métodos para guardar el índice y los módulos de los documentos, así como la estructuración del mismo en los directorios.
Los métodos implementados en el constructor del índice son los siguientes:

- init (String indexPath): Este método simplemente inicializa los distintos valores de la clase, actuando como constructor de la misma.
- save (String indexPath): Este método guardara el diccionario de términos una vez serializado.
- indexText (String text, String path): Este método crea una lista que contiene todos los términos del documento y actualiza o crea PositionalPosting para después añadirlos al diccionario.

Por último la clase PositionalIndex simplemente carga el índice y el diccionario serializados para poder realizar búsquedas posteriormente. La mayor parte de los métodos de esta clase ya están implementados en BaseIndex. El unico metodo que hemos creado ha sido el constructor de la clase que se encarga de cargar el PositionalDictionary guardado e incluirlo al índice.

Pregunta 3

Para probar el crawler será necesario crear una carpeta test a la altura de la carpeta src. Dentro de esta carpeta será necesario especificar un fichero llamado “semillas.txt”, donde se escribirán las urls que se van a analizar. Una vez ejecutado el crawler obtendremos los resultados tanto en un fichero dentro de la carpeta test llamado urlsWebCrawler.txt, donde se guardaran todas las urls encontradas. También se almacenarán las urls dentro de la carpeta graph, en el fichero webgraph.dat.

En nuestro caso nuestro fichero de semillas tiene una url:

- “https://en/wikipedia.org/wiki/Estrella_Galicia”

Hemos elegido estas urls, porque a pesar de que nuestro crawler extrajese la información correctamente, hemos tenido muchos errores a la hora de crear el índice de lucene con el resultado del análisis, creemos que esto se debe a que al traer de cargar páginas cuyo contenido es dinámico y se va actualizando muchas veces al dia, muchos de los enlaces expiran, por lo que aparecen gran cantidad de errores. Alguno de estos errores han sido los siguientes:

- https://www.elmundo.es/ : Tiene páginas que dan error 404, las que están destinadas a suscripción.
- https://www.eldiario.com/ : Tiene páginas que dan error 404.
- https://stackoverflow.com/: Problemas con certificados de la página.

Además de estos errores hemos tenido que hacer algunos cambios en el crawler, como no permitir que se guarden páginas con protocolo http, ya que estas no sirven para crear un índice con lucene. La clase que prueba el crawler se encuentra en el modulo “es.uam.eps.bmi.search.test”. Los datos del índice construido se guardarán en una carpeta test_out.

Una posible mejora del crawler que hemos implementado sería comprobar la existencia de las urls que se guardan, ya que gran parte de los errores que hemos encontrado al probar este crawler es que muchos de los enlaces hayan expirado, que los servidores que los hostean no den respuesta… De esta manera al crear el indice no tendriamos ese tipo de problemas.

Adjuntamos la carpeta test junto con el fichero semillas.txt para poder probar el funcionamiento del crawler.