Memoria práctica 3

Búsqueda y minería de la información

Jorge Cifuentes

Alejandro Martín

**Ejercicios realizados**: todos a excepción del último.

**Comentarios relevantes:**

**Ejercicio 1**

Hemos implementado la búsqueda proximal con el algoritmo visto en teoría. Busca las comillas en la query para decidir si hacer búsqueda literal.

**Ejercicio 2**

Para hacer la búsqueda proximal, tenemos dos clases:

*PositionalIndexBuilderImp,* que usa:

PositionalDictionary (diccionario de PositionalPostingsList) para la creación; es una lista de PositionalPostingImpl.

*PositionalIndexImpl*, que usa:

PositionalDiskHashDictionary: Como DiskHashDictionary, pero carga también la lista de posiciones al obtener un posting.

Nos hemos basado en el índice de disco para esta implementación.

**Ejercicio 3**

Para el PageRank, hemos usado dos mapas hash de paths del documento: una docID-path y otra path-docID, ya que nos parecía que su coste en memoria merecía la pena respecto a solo tener el primer mapa, y buscar docID a partir de path mediante una búsqueda lineal.

Para obtener el score, realizamos las iteraciones necesarias, parando en dos casos: si se llega al límite pasado por parámetro, o si, para todos los documentos, la diferencia entre el PageRank de esta iteración y el de la anterior es menor que cierto umbral definido como constante en la clase. Esto hace que tarde menos, y dependiendo de los ceros de la constante de umbral, los valores serán más o menos parecidos a no hacer dicha comprobación, manteniéndose igual la ordenación.

En cuanto a los nodos sumidero comprobamos, a partir de los links obtenidos de archivo, si hay sumideros. Si hay, ajustamos todos los PageRanks con la fórmula . Esta fórmula es una simulación de agregar un enlace a todos los nodos, a cada nodo sumidero, lo cual sería muy costoso de hacer en memoria.

**Ejercicio 4**

Hemos programado un crawler muy básico que en sucesivas iteraciones explora páginas. Para ello, primero se descarga el archivo *robots.txt*, añadiendo las páginas *Disallow* a una lista. Después, mediante Jsoup, obtenemos el contenido de dicha página y sus enlaces, que añadimos a la frontera de crawling. Así vamos explorando la web.

De manera paralela, guardamos cada documento con el IndexBuilder pasado por parámetro, y creamos un archivo de grafos (from -> to) que puede ser usado por PageRank.

Para **ejecutar**, hemos usado un TestCrawler.java, que se basa en el PositionalIndexBuilderImpl.

**Ejercicio 5**

minmax