Descripció d'estructures i algorismes utilitzats

En el nostre programa hem utilitzat una base de dades SQLite JDBC on hi hem emmagatzemat els documents, autors i usuaris que hem inserit.

Els documents poden estar estructurats en una List de documents anomenada DocumentsSet o bé un TreeMap<Double,List<Document>> on el double és el pes de la llista de documents que hi ha després anomenada SortedDocumentsSet. Globalment, tenim la classe DocumentsCollection que és una List que conté tots els documents de la base de dades i en controla el pes de les paraules calculat amb l'algorisme tf-idf.

Els autors, igual com els documents, també estan inclosos a una classe anomenada AuthorsCollection que és una List d'autors.

La cerca dels documents d'un autor recorre tota la col·lecció de documents linealment ajudant-se de la funció stream() per a distingir els noms dels autors.

La cerca dels documents d'un usuari recorre tota la col·lecció de documents linealment ajudant-se de la funció stream() per a distingir els títols dels documents.

La cerca d'un autor donat un prefix també recorre tota la col·lecció d'autors linealment ajudant-se de la funció stream() per a distingir els títols dels documents.

La cerca d'un document donat un títol i un autor es realitza amb una cerca lineal a la col·lecció de documents.

La cerca de 'k' documents similars afegeix a un SortedDocumentsSet -limitat al mínim de 'k' i la mida de la col·lecció de tots els documents- tots els documents de la col·lecció, eliminant els que tenen el valor de rellevància més petit cada vegada que s'hi insereixi un document que no hi cap.

La cerca per query de 'k' documents similars converteix la query a un document i llavors aquest s'utilitza per cridar a la cerca de 'k' documents similars per obtenir el resultat amb un SortedDocumentsSet.

Per a determinar la rellevància d'un document, hem utilitzat un algorisme que es diu tf-idf (term frequency-inverse document frequency). Aquest algorisme aplica pesos a les paraules de manera que una paraula té més pes si està molt repetida dins del seu document però poc en el conjunt total de documents.

Per la cerca booleana, s'ha usat un arbre amb un operant per a cada node amb excepció de les fulles, que contenen una paraula o una seqüència d'aquestes.
L'expressió booleana es tradueix en un arbre n-ari d'aquest tipus per després poder contrastar cada un dels documents amb aquest arbre.

Per aquesta segona part, determinarem quins documents compleixen les condicions especificades a l'arbre mitjançant una estructura formada a patir de totes les paraules de cada document consistent en un Map<String, Map<Integer, Set<Integer>>> de tal manera que pots buscar una paraula, obtenint el conjunt de nombres de frase en que aquesta apareix i dintre d'aquest conjunt, cada frase on apareix aquesta paraula té un conjunt de nombres indicant la posició exacte d'aparició dintre d'aquesta frase.
D'aquesta manera es pot comprovar quines frases d'un document compleixen les condicions de cada fulla i anar aplicant recursivament els operadors als conjunts de frases que compleixen les condicions dels fills.