Estructura de dades i Algorismes Utilitzades

Algorismes

Tf-idf:

És una estadística numèrica que pretén reflectir la importància d'una paraula per a un document d'una col·lecció o corpus. El valor tf–idf augmenta proporcionalment al nombre de vegades que apareix una paraula al document i es compensa amb el nombre de documents del corpus que contenen la paraula, cosa que ajuda a ajustar-se al fet que algunes paraules apareixen amb més freqüència en general. El tf és La freqüència del terme és la freqüència relativa del terme t dins del document d, i el idf és La freqüència inversa del document és una mesura de quanta informació proporciona la paraula.

Fem servir aquest algorisme per Consultes K documents més rellevants i K documents més semblants.

Cosinus Similarity:

La semblança del cosinus és una mesura de semblança entre dues seqüències de nombres. Per definir-lo, les seqüències es veuen com a vectors en un espai interior del producte, i la semblança del cosinus es defineix com el cosinus de l'angle entre elles, és a dir, el producte escalat dels vectors dividit pel producte de les seves longituds. La semblança del cosinus sempre pertany a l'interval [-1,1]

Fem servir aquest algorisme per calcular semblança dels documents.

Estructura de dades

Trie:

Un Trie és un arbre ternari on cada node conte un caràcter i pot ser final o no. Està ordenat de forma que el valor del node esquerre és menor al de l'arrel i el valor del node dret és més gran que el de l'arrel. El node central no té cap relació d'ordre amb el seu pare.

A partir d'aquesta estructura es poden formar paraules seguint aquestes normes. Començant des de l'arrel del Trie, podem baixar per un dels 3 fills. Si baixem pel central, agafem el valor del seu pare i ens guardem el caràcter. Altrament, si baixem pel fill esquerre o dret, no fem res. Continuem recorrent el Trie afegint els caràcters quan baixem pel node central. Per poder parar, només hem de trobar un node que sigui final.

Fem servir aquesta estructura de dades per guarda noms dels autors per facilitar la consulta PrefixAutor, ja que per buscar tots els autors amb un prefix només s'ha de recórrer el Trie fins a arribar a un node que "contingui" aquell prefix (que afegint els caràcters obtenim el prefix).

A partir d'aquest node només hem de buscar els possibles sufixos que es poden formar (camins fins a nodes finals) per trobar tots els autors amb el prefix.

Això té un cost log(h), h és l'alçada de l'arbre.

Binary tree:

És un arbre binari modificat de forma que cada node compta amb un valor que és un String i un booleà que indica si el node està negat o no.

L'utilitzem per poder representar una expressió booleana com un "Expression Tree" on els nodes interns tenen com a valors operands lògics ('&' o '|') i les fulles són àtoms (paraules o frases). Per tal d'avaluar l'expressió l'únic que s'ha de fer és:

Primer avaluar la part esquerra i avaluar la part dreta recursivament (obtenint els documents que la compleixen).

Finalment, només s'ha de fer una operació entre els dos conjunts, una unió (si l'operador del node pare és un '|') o bé la intersecció (si l'operador del node pare és un '&').

HashMap:

El HashMap és una implementació d'un map utilitzant una taula de hash. No garanteix l'ordre del mapeig. El seu avantatge és quan fem servir get, put, remove el cost és O(1).

TreeMap:

El TreeMap és una implementació d'un map utilitzant un red-black tree. Aquesta implementació ens garanteix l'ordre del mapeig. El desavantatge es que per fer un get, put o remove té un cost O(log n).

HashSet:

El HashSet és una implementació d'un set utilitzant una taula de hash. No garanteix l'ordre del mapeig. El seu avantatge és quan fem servir get, put, remove el cost és O(1).

TreeSet:

El TreeMap és una implementació d'un set utilitzant un red-black tree. Aquesta implementació ens garanteix l'ordre del mapeig. El desavantatge és que per fer un get, put o remove té un cost O(log n).

ArrayList:

És una implementació d'un array, però que es pot redimensionar i a part té molts mètodes útils per a fer més senzill el codi.