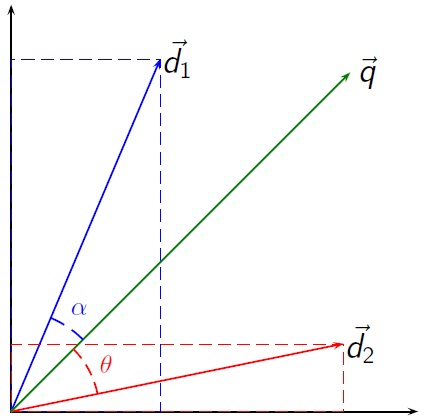
Algoritmul de cautare dupa continut

Pentru a compara doua texte dupa continut vom folosi o reprezentare vectoriala a textelor. Modelul vectorial de reprezentare a textelor este un model algebric prin care putem reprezenta orice text ca un vector de dimensiunea alfabetului din care pot aparea cuvinte. Odata ce am reprezentat documentele continute in baza de date si stringul de cautare ca si vectori putem sa calculam unghiul dintre vectorul fiecarui document si a stringului de cautare. Respectiv documentele care vor avea vectorii mai apropiati de vectorul stringului de cautare (diferenta dintre unghiul vectorului de query si a documentului va fi cit mai mica) vor fi mai relevante, asemanindu-se mai mult.

Exemplu:



In aceasta imagine putem vedea vectorii reprezentativi pentru un oarecare document d1, documentul d2 si stringul de cautare q. Dupa cum α<θ documentul d1 este mai asemanator cu stringul de cautare decit documentul d2.

In practica este mai simplu sa calculam *cosinusul* unghiurilor decit valorile propriuzise ale lor.

In cazul din desen avem:


\cos{\theta} = \frac{\mathbf{d_2} \cdot \mathbf{q}}{\left\| \mathbf{d_2} \right\| \left \| \mathbf{q} \right\|}


Unde d2\*q reprezinta produsul punctual dintre vectorul d2 si q ( se inmulteste fiecare element de pe pozitia i din d2 cu fiecare element de pe pozitia i din q), iar \left\| \mathbf{d_2} \right\|  este norma vectorului d2, respectiv q. Mai exact:


\left\| \mathbf{q} \right\| = \sqrt{\sum_{i=1}^n q_i^2}


Calcularea vectorilor caracteristici si determinarea valorii cosinusului unghiului dintre vectorii respectivi:

Fie un document D din baza de date si stringul de cautare Q. Notam cu d=(wd1,wd2,wd3,…,wdn) vectorul documentului D si q=(wq1,wq2,wq3,…,wqn) vectorul stringului de cautare. Pentru a calcula valorile wd1..wdn, wq1..wqn vom aplica metoda tf-idf (term frequency-invers document frequency) propusa de chinezii Wong si Yang. Conform acestei scheme avem:


w_{t,d} = \mathrm{tf}_{t,d} \cdot \log{\frac{|D|}{|\{d' \in D \, | \, t \in d'\}|}}


Unde:

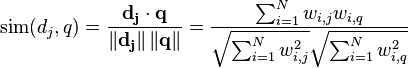
* - frecventa de aparitie a cuvintului t in documentul d (sau in stringul de interogare q);

|D| - numarul total de documente in multimea de documente (in cazul nostru numarul total de intrebari sau raspunsuri din baza de date);

|\{d' \in D \, | \, t \in d'\}| - numarul de documente in care apare cuvintul t.

Respectiv vom memora intr-o tabela separata din baza de date pentru fiecare cuvint din alfabet (care poate creste de fiecare data cind este dat un raspuns sau se posteaza o intrebare) de cite documente distincte este continut cuvintul respectiv. Pentru ca interogarile specifice acestei tabele sa fie eficiente vom crea un index dupa coloana in care vom tine cuvintele ( sau le vom incarca intr-un trie in momentul pornirii serverului ). La fel din motive de eficienta vom introduce in tabel doar cuvintele cheie ( cuvintele mai lungi de 4-5 litere ) pentru a evita procesarea inutila a tot felul de prepozitii si chestii de genul. Inca o optimizare ar fi sa tinem vectorii caracteristici ca perechi (cheie,valoare) unde cheia va fi pozitia iar valoarea este w\_cheie, astfel vom pastra doar valorile nenule din vectori economisind in mediu multa memorie si timp intrucit documentele noastre nu contin prea multe cuvinte in timp ce alfabetul poate fi destul de mare ( de ordinul 10^5-10^6 ) in mediu.

Dupa ce am calculat vectorii caracteristici calculam unghiul dintre document si query conform formulei cosinusului:



Unde cu ***sim*** am notat similaritatea ( cos α ).

Deoarece multe documente ar putea semana farte putin cu stringul de interogare, vom returna de fiecare data min(50, |D| ) rezultate. Pentru a mentine eficient lista de elemente sortata vom tine un min\_heap pe care il vom popula cu maxim 50 elemente si il vom reactualiza permanent.

La sfirsit vom afisa rezultatele gasite sortate desrescator in functie de numarul de voturi.