Cerca i Anàlisi d'Informació Massiva

Laboratori: Sessió 2

Programming with ElasticSearch

Sergi Faz

Oriol Vidal

17/10/2019

Introducció

L'objectiu d'aquesta pràctica és estudiar i entendre com funcionen els mecanismes que ajuden a reduir el norme de termes dels documents que indexem, com per exemple, aquells termes que no són inteligibles o no aporten informació rellevant pel contingut del mateix document.
El primer pas d'aquest procés consisteix en convertir el text en tokens. Després, també podem normalitzar les cadenes obtingudes i filtrar els tokens que han quedat vàlids però que no són útils.

Aquesta segona sessió l'hem dividir en dues parts. A la primera hem realitzat un estudi sobre diferents col·leccions de text, utilitzant *ElasticSearch*, aplicant diferents tokens, filtres i algoritmes de filtratge.

A la segona part, hem implementat/completat una sèrie de funcions que ens permeten calcular la similaritat entre dos documents, basant-nos amb la informació que tenim d'aquests un cop els hem processat i indexat.

Estudi

Explicació

A continuació, farem una petita descripció de cada token que hem utilitzat en l'estudi, perquè quedi clar quin és el seu funcionament.

- **Standard:** proporciona una tokenització basada en gramàtica i funciona bé per a la majoria dels idiomes.
- **Letter:** divideix el text en termes sempre que es troba amb un element que no és una lletra. Fa un treball raonable per a la majoria de les llengües europees, però fa un treball terrible per a algunes llengües asiàtiques, on les paraules no estan separades per espais.
- **Whitespace:** divideix el text en termes sempre que tingui un caràcter en blanc.
- **Classic:** té heurístiques per al tractament especial de les sigles, noms de l'empresa, adreces de correu electrònic i noms d'amfitrió d'Internet. Tot i això, aquestes regles no sempre funcionen i el testimoni no funciona bé per a la majoria de llengües diferents de l'anglès.

Comparacions

Primer de tot hem comparat els tres algoritmes de filtratge que ens ofereixen amb el token *Standard*, i hem vist que *Snowball* és l'ideal per realitzar l'estudi. Un cop triat l'algoritme amb el que analitzarem els textos, hem creat diferents índexs a partir de les diferents combinacions entre tokens i filtres per tal d'averiguar quins són els més útils, quins els més restrictius i quins els més "permissibles".

Aplicant els tres filtres dels que disposem, ens hem donat compte que la diferencia no era considerable, per tant, el que marcarà la diferència a l'hora de comptar les paraules dels textos, és el token.

Els resultats següents s'han obtingut mitjançant l'script *count.sh*, adjuntat amb l'output *outCount.txt*.

Token	Filter	Algorithm	#words
Standard	Lowercase	Snowball	104314
Standard	Lowercase	Porter_Stem	107191
Standard	Lowercase	Kstem	111772
Standard	Lowercase + Ascii	Snowball	104302
Standard	Lowercase + Ascii + Stop	Snowball	104292
Letter	Lowercase + Ascii + Stop	Snowball	70754
Whitespace	Lowercase + Ascii + Stop	Snowball	271713
Classic	Lowercase + Ascii + Stop	Snowball	107088

Figura 1: Taula de comparacions de índexs

Conclusions

Com es pot observar a la *Figura 1*, el token més estricte és el *Letter*, donat que només es queda amb les lletres, tot el demés ho elimina. Per altre banda, el whitespace és el més permissiu, ja que només separa el text en espais en blancs i ho permet tot. Entre els dos restants, veiem que el nombre de paraules és molt semblant. Per tant, concluïm aquest petit estudi, afirmant que els tokens ideals per al càlcul de similitud de documents són el *Standard* i el *Classic*, aplicant els tres filtres *Lowercase*, *Ascii* i *Stop*, amb un algoritme *Snowball*, ja que amb ells obtenim resultats propers a la mitjana dels resultats obtinguts a les quatre últimes files.

Pràctica

Explicació

Per a poder fer les proves que es demanen per aquesta pràctica, hem hagut d'utilitzar l'script *TFIDFViewer.py*, que calcula la la similitud de dos documents modelats amb l'esquema **tf-idf**.
Aquest càlcul es realitza concebent els documents com a vectors de termes, i conseqüentment la col·lecció de documents com una matriu *termes* * *documents*.

D'aquesta manera, per cada terme, podem saber a quants i quins documents es troba, i les vegades que apareix a cadascun d'aquests.

Per a calcular la similitud de dos documents, a partir d'ara *sim(d1, d2)*, utilitzem el càlcul del cosinus de l'angle que formen els vectors que representen *d1* i *d2*. Aquesta expressió es representa de la forma:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

En aquest cas, els vectors que representen *d1* i *d2* són vectors de pesos dels termes que contenen cada document(a partir d'ara es parlarà de *d1* i *d2* com als vectors de pesos dels documents).

L'esquema **tf-idf** és utilitzat per a poder obtenir aquests dos vectors de pesos. Cada pes d'un terme, s'expressa de la forma $w_{d,i} = tf_{d,i} * idf_i$, on $tf_{d,i}$ és la freqüència del terme *i* en el document *d*, i idf_i (inverse document frequency) ens servirà per modelar el pes del terme en funció de la freqüència que té aquest en tota la col·lecció de documents. Així doncs, com més freqüent és un terme en un document ($tf_{d,i}$), més pes ha de tenir; per altra banda, com més freqüència tingui el terme dintre tota la col·lecció, la seva freqüència dins d'un document és menys rellevant, de manera que el seu pes ha de disminuir en tota la col·lecció de documents(idf_i). Aquestes són les expressions dels dos conceptes descrits:

$$tf_{d,i} = \frac{f_{d,i}}{max_j f_{d,j}}$$

on $f_{d,i}$ és el nombre de vegades que apareix el terme al document, i $max_j f_{d,j}$ és el nombre de vegades que apareix el terme més freqüent del document.

$$idf_i = log_2 \frac{D}{df_i}$$

on *D* és el nombre total de documents, i df_i és el nombre de documents que contenen el terme.

Experimentació

Un cop tenim ben clar el què s'ha de fer, cal acabar d'implementar l'script *TFIDFViewer.py* per a què ens calculi *sim(d1, d2)* de dos documents d'una mateixa col·lecció indexada per l'*Elastic Search*.

El funcionament d'aquest script consta de tres parts: cercar els documents que s'indiquen a l'índex corresponent; obtenir els vectors de pesos de cada document; i calcular la seva similitud.

El primer pas consta de fer la *query* del document al *ElasticSearch*, el qual té un atribut on hi ha el path exacte. Hem vist que a l'script *IndexFilesPreprocess.py*, abans de tokenitzar cada document, guarda un camp al seu espai de l'índex amb el seu path absolut, de manera que el podem trobar a l'índex fàcilment.

El segon pas correspon a la funció *toTFIDF*, la qual retorna el vector de pesos (normalitzat) de cada document que se li passa com a paràmetre, que posteriorment seran utilitzats per a calcular la similitud dels dos documents donats. Per a obtenir aquests dos vectors, comença calculant el nombre de vegades que apareix cada terme del document al document, de manera que obté una llista de parells (*terme*, *#aparicions*), on no hi ha cap valor del camp *#aparicions* igual a 0. També calcula, per a cada terme, el nombre de documents de la col·lecció on apareix. Finalment calcula el nombre total de documents de la col·lecció.
A partir d'aquí nosaltres hem implementat els càlculs per obtenir el pes de cada terme (llista de (*terme*, *pes*)) seguint l'esquema **tf-idf** ja explicat. Per la funció *toTFIDF* només hem afegit les línies 93:97, i hem implementat també la funció *normalize*, la qual calcula el vector unitari dels pesos, mantenint però l'estructura (*terme*, *pes*).

Finalment, hem implementat la funció *cosine_similarity*, que calcula el resultat final. Al haver d'operar amb llistes de diferents tamanys, on no hi ha termes amb pes igual a 0, el càlcul del producte dels dos vectors s'ha fet quan, iterant sobre les dues llistes, els termes coincidien.

Proves

Per provar la correctesa de l'script, primer l'hem executat comparant dos documents idèntics, obtenint així una similaritat exactament igual a 1. Ho hem fet amb el grup de documents *novels*, i el document *DarwinOriginofSpecies.txt*.
Després hem volgut comprovar que la similitud entre documents del mateix tipus és major a la similitud de dos documents de termes completament diferents. Això és, amb el grup de documents *20_newsgroups*, si la similitud entre documents del mateix subtipus de notícia, és major que la similitud entre documents que formen part de diferents subgrups.

Totes les proves s'han realitzat mitjançant l'script sim.sh, adjuntat juntament amb el seu output, similarities.txt.
Només especifiquem el token, ja que totes les proves comparteixen el filtrat que hem especificat a la conclusió de la primera part de la pràctica.

Documents	Standard	Classic	Letter	Whitespace
alt.atheism/0000000 alt.atheism/0000446	0.00708	0.00000	0.00006	0.00446
alt.atheism/0000338 sci.space/0014451	0.00056	0.00059	0.00055	0.00044

Figura 2: Similituds obtingudes aplicant els diferents tokens.

Observant la *Figura 2* podem comprovar que els tokens *Standard* i *Whitespace* són els que han donat els resultats esperats. Hem volgut comprovar si utilitzant els altres tokens amb un algorisme de filtrat menys agressiu ens podien donar millor resultats, i tampoc ha sigut així (veure *Figura 3*). Hem fet un altre script *sim2.sh* i l'hem adjuntat juntament amb els seus resultats al fitxer *similarities2.txt*.

Documents	Classic	Letter
alt.atheism/0000000 alt.atheism/0000446	0.00000	0.00000
alt.atheism/0000338 sci.space/0014451	0.00055	0.00051

Figura 3: Similituds obtingudes aplicant l'algorisme Kstem per filtrar després d'aplicar els tokens Classic i Letter.

Conclusions

Podem dir que, a pesar de creure en algun moment que els token ideals són *Standard* i *Classic* basant-nos només amb la quantitat de paraules que deixen als documents, no ha sigut així. Els resultats s'han basat amb el tractament que fan els tokenizers sobre els caràcters dels documents.
El tractament que fan els tokenizers *Classic* i *Letter* modifica les paraules del documents, de manera que costa trobar resultats precisos, a diferència del *Standard*, que fa una separació basant-se amb l'estructura gramatical. Utilitzar *Whitespace* també ens ha donat bons resultats ja que és molt permissiu.