Facultad de informática de Barcelona

Cerca i Anàlisi d'Informació Massiva

Session 2: Programming with ElasticSearch

07/10/2022

Tardor 2022-2023

Marc Camarillas Parés
Marc Nebot Moyano

Marc Camarillas Parés
Marc Nebot Moyano

ÍNDEX

1. Modifying ElasticSearch index behavior **3**

2. Computing tf-idf's and cosine similarity **4**

2.1 Experimentació 4

3. Conclusions **5**

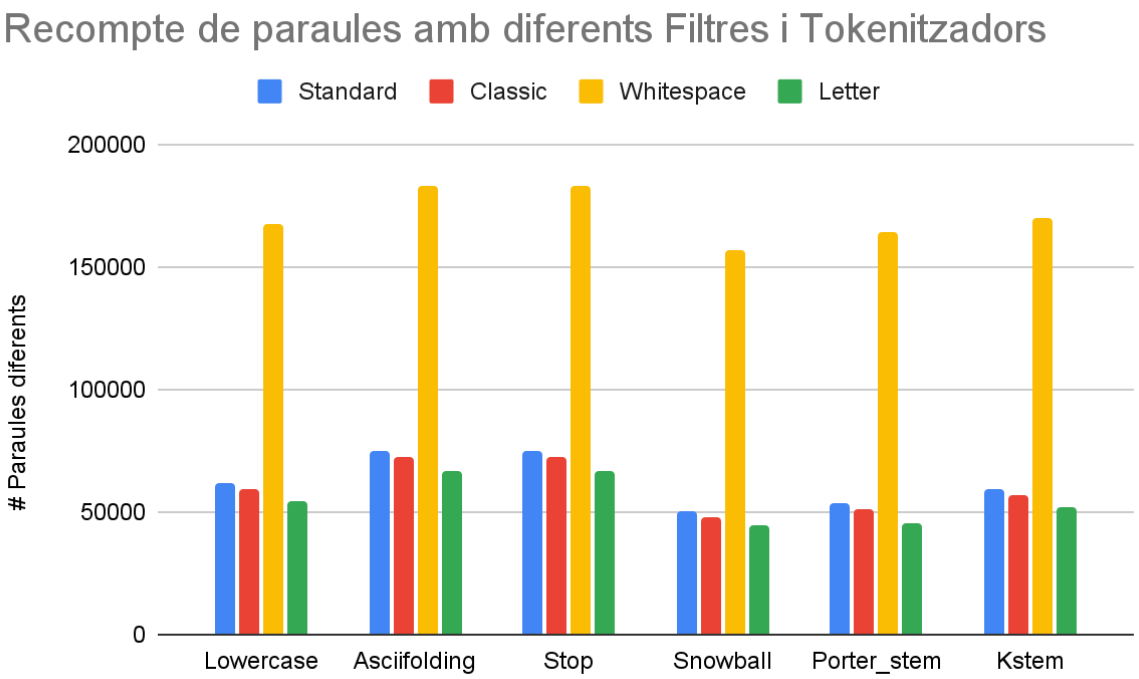
Marc Camarillas Parés
Marc Nebot Moyano

1. Modifying ElasticSearch index behavior

Per tal de comprovar quin és el tokenitzador i el filtre menys permissiu, hem comptat quantes paraules diferents surten de cada parella tokenitzador-filtre.

	Standard	Classic	Whitespace	Letter
Lowercase	61825	59569	167063	54455
Asciifolding	74813	72483	183188	67106
Stop	74803	72476	183155	67099
Snowball	50431	48018	157076	44872
Porter_stem	53498	51085	164486	45523
Kstem	59491	57122	169951	51685

Taula 1: Resultats amb diferents filtres i tokenitzadors



Com podem veure tant en la gràfica com en la taula el tokenitzador més agressiu és el de letter i el filtre més agressiu és el snowball, ja que són els que menys paraules accepten. També, podem afegir més d'un filtre a la vegada, sent aquesta una manera d'obtenir encara un filtratge més dur que el obtingut.

Marc Camarillas Parés
Marc Nebot Moyano

Hem comprovat que l'ordre de filtratge és bastant important ja que si primer apliquem stop i després snowball no ens dóna el mateix nombre de paraules que si apliquem primer snowball i després stop.

2. Computing tf-idf's and cosine similarity

En aquest apartat haurem de comprovar les similituds de diferents documents. Per aconseguir realitzar la experimentació hem modificat l'script adjuntat en el zip, on hem afegit la similitud del cosinus normalitzada per tal de comparar adequadament els documents.

2.1 Experimentació

En primer lloc, hem procedit a comparar els document d'Alt Altheism amb sí mateixos per observar si havien similituds al ser del mateix gènere obtenint els següents resultats:

DOCS	0000000	0000001	0000002	0000003
0000000	1,00000	**0,25966**	0,05135	0,05275
0000001	**0,25966**	1,00000	0,11078	0,10164
0000002	0,05135	0,11078	1,00000	0,03574
0000003	0,05275	0,10164	0,03574	1,00000

Taula 2: Resultats de comparar *Alt Altheism* amb si mateix

Podem observar tal i com esperàvem que un document entre sí mateix dóna una similaritat d'1, mentre que entre altres documents del mateix gènere aquesta similaritat és bastant baixa tenint un màxim de similitud del 0,25966.

Seguidament compararem la similaritat del gènere *Alt Altheism* amb *Sci Space* per tal d'observar com varia els resultats si es comparen diferents gèneres:

DOCS	0000000	0000001	0000002	0000003
0014000	0,04855	0,06515	0,03109	0,01446
0014001	0,03914	0,02230	0,01333	0,00828
0014002	0,06404	**0,07916**	0,05585	0,02002
0014003	0,07582	0,04283	0,02734	0,02761

Taula 3: Resultats de comparar documents de *Sci Space (eix vertical)* amb *Alt Altheism (eix horitzotal)*

Marc Camarillas Parés
Marc Nebot Moyano

En aquesta taula es veu clarament que no s'assemblen gens els dos documents tenint un màxim de similitud del 0,07916, per tant, sí que existeix una diferència bastant gran entre ambdós gèneres comparats.

Finalment, per concloure les comparacions hem decidit comparar *Sci Space* amb si mateix per veure que realment existeixen grans diferències entre gèneres però que un gènere amb si mateix és bastant més similar, obtenint els següents resultats:

DOCS	0014000	0014001	0014002	0014003
0014000	1,00000	0,01557	0,05774	0,07409
0014001	0,01557	1,00000	0,10299	0,10299
0014002	0,05774	0,10299	1,00000	**0,26139**
0014003	0,07409	0,05552	**0,26139**	1,00000

Taula 4: Resultats de comparar *Sci Space* amb si mateix

Aquesta taula ens dóna uns resultats similars als que ens han donat quan hem comparat *Alt Altheism* amb si mateix. Podem comprovar que un document amb si mateix ens segueix donant una similitud de 1, mentre que la màxima similitud entre documents de *Sci Space* és de 0,26139, bastant més alta que la comparació amb diferents gèneres cosa que té sentit ja que pertanyen al mateix gènere i els temes teòricament haurien d'estar més o menys relacionats.

3. Conclusions

D'acord amb els resultats, té sentit que snowball sigui el filtre més estricte, ja que agrupa un conjunt de paraules del mateix significat en una sola paraula fent que es redueix considerablement el nombre de paraules. Per altre banda també és coherent que el nombre de paraules sigui diferent segons l'ordre ja que quan apliquem snowball pot donar-se el cas que més d'una paraula s'agrupi en una stop word i després s'elimini, en canvi al revés primer s'eliminaria la stop word i després s'agruparien les paraules en aquella stop word fent que el nombre total de paraules sigui superior. Amb lowercase y stopword passa el mateix.

En la part de la implementació del TFIDFViewer, no hem tingut gaire complicacions tret de la part de calcular el cosine similarity. ja que inicialment només vam tenir en compte vectors de mides iguals, i vam trigar una mica en comprendre que fallava, per el demés, vam entendre-ho tot a la primera i no va ser cap complicació.

Marc Camarillas Parés
Marc Nebot Moyano

Gràcies a aquesta pràctica hem aprofundit més en les maneres de preprocessar els textos, aprenent el funcionament i significat dels diferents filtres i tokenitzador, mentre que ens ha ajudat a comprendre més el funcionament de tf-idf i la similitud del cosinus explicats a classe.

Finalment respondrem a les següents preguntes que se'ns planteja:

1. Now you can answer the question, what word is the most frequent one in the English language?

Hem provat de comptar les paraules que apareixen en el conjunt de fitxers arxiv abs amb el tokenitzador i el filtre més agressiu (letter i snowball) obtenint com a resultat la paraula ***the*** amb un total de 985950 ocurrencies sent 1.5 vegades més que la segona paraula, ***of*** amb un total de 623014.

Si apliquem un filtre stop per eliminar les stop word obtenim que la paraula més utilitzada és, we amb un total de 108525 aparicions.

2. A final question, have you noticed that we are searching the documents using the path name? By default, all text fields are tokenized. Yet, if the path field had been tokenized, these searches would not succeed, right? What did we do differently when indexing the documents so we can look for an exact match in the path field?

El que està passant al indexar els fitxers es que es guarda el path en un llista i s'agrega el amb les '/' tal i com podem veure en la següent línia de codi →`lfiles.append(lf[0] + '/' + f)`