Sessió 2: Programes amb l'Elasticsearch

Dani Beltrán i Toni Cifre

October 2020

Contents

1 Efectes sobre l'index **3**

2 Python files **3**

2.1 Ús del programa 3

2.2 Canvis i addicions 3

2.3 Observacions addicionals 4

1 Efectes sobre l'index

Al realitzar la tokenitzaciò de paraules de diverses maneres (separant per espais en blanc, caràcters que no son lletres, etc..) ens resulta l'augment de paraules que ja es trobaven al nostre sistema (mots compostos). Conjuntament amb el filtratge que hem realitzat posteriorment on hem normalitzat tot el text (passant-ho tot a minúscules) i hem realitzar un *stemming* (extreure d'un terme la seva arrel i guardar ho) hem pogut evitar moltes paraules derivades d'una mateixa (per exemple, pont, pontet, pertanyen a una mateixa arrel pont) reduint aixi la mida del index a més d'una possible disminució dels mots amb faltes ortogràfiques (sobretot al final de la paraula).

Gracies a tot aixo ens ha permès treballar en indexes molt més petits (hem passat d'utilitzar un index de 61.000 mots a 40.000), hem dificultat trobar termes nous que no reconeguem (donat que al treballar en arrels, abastem totes les paraules derivades de aquesta) i hem facilitat la similitud entre diversos indexes.

2 Python files

2.1 Ús del programa

El fitxer `TFIDFviewer.py` ens permet calcular l'esquema dels pesos *tfidf* per un index i un document, quant de rellevant es aquest text en aquest index. A més amb el calcul de la similitud, ens permet que donat dos textos d'un mateix conjunt calcular quant s'apareixen entre si (prenent 1 com la similitud màxima, iguals, i 0 la similitud mínima).

2.2 Canvis i addicions

En la selecció dels 3 documents *Python* de la pràctica, només es necessari modificar el document `TFIDFviewer.py`, tot i que si es vols experimentar un poc més amb els filtres i tokenització dels índexs es poden afegir més opcions al document `IndexFilesPreprocess.py` per experimentar amb filtres més avançats. En el cas del document `TFIDFviewer.py`, hem modificat diferents funcions i n'hem creat d'altres. En primer lloc, a la funció *toTFIDF(client, index, file_id)* hem calculat *tf* i *df* per cada un dels documents, i a continuació retornem el vector normalitzat.

Per a normalitzar aquest vector, utilitzem la funció que hem afegit, *normalize(tw)*, la qual, en primer lloc calculem la norma 2 del vector, donat com a paràmetre *tw*, i procedirem a la normalització d'aquest amb la divisió de tots els seus termes. Amb aixo hem aconseguit que tot valor de l'array sigui com a màxim 1 i com a mínim 0, evitant aixi que a més gran el document més similar pot resultar al comparàs amb diferents conjunts.

La funció *print_term_weight_vector(file1_tw)* tan-sol l'hem modificat per a poder mostrar per consola de forma ordenada i llegible cada un dels termes amb els pesos obtinguts prèviament a traves del vector passat per paràmetre.

Per últim, hem modificar la funció *cosine_similarity(tw1, tw2)* per a que calculi i retorni la similitud entre dos vectors on els termes estan ordenats alfabèticament i prèviament normalitzat. Per a poder calcular el producte escalar, hem creat la funció *intersect(l1, l2)* on retorna el la multiplicació dels pesos on els termes coincideixen a partir de dos vectors passats per paràmetre, i la nostra funció de *cosine_similarity(tw1,tw2)* retornara el sumatori del vector retornat, es a dir, el producte escalar dels dos vectors passats per paràmetre.

2.3 Observacions addicionals

Per poder realitzar la verificació del nostre fitxer modificat hem procedit a realitzar les comprovacions amb els documents i l'index dels exercicis realitzats a classe, on ja tenim calculats prèviament els valors.

Un cop introduït l'index de manera manual (cada document nou en el seu fitxer corresponent) sense haver aplicat stemming (ja que als exercicis no l'hem realitzat) hem procedit a l'execució del nostre programa.

Els càlculs realitzats eren correctes (*sim(d3,d4) = 0.035* per els dos casos), aixi que ja hem comprovat que el nostre programa era correcte.