Laboratori de CAIM

Pràctica 2

Walter J.Troiani

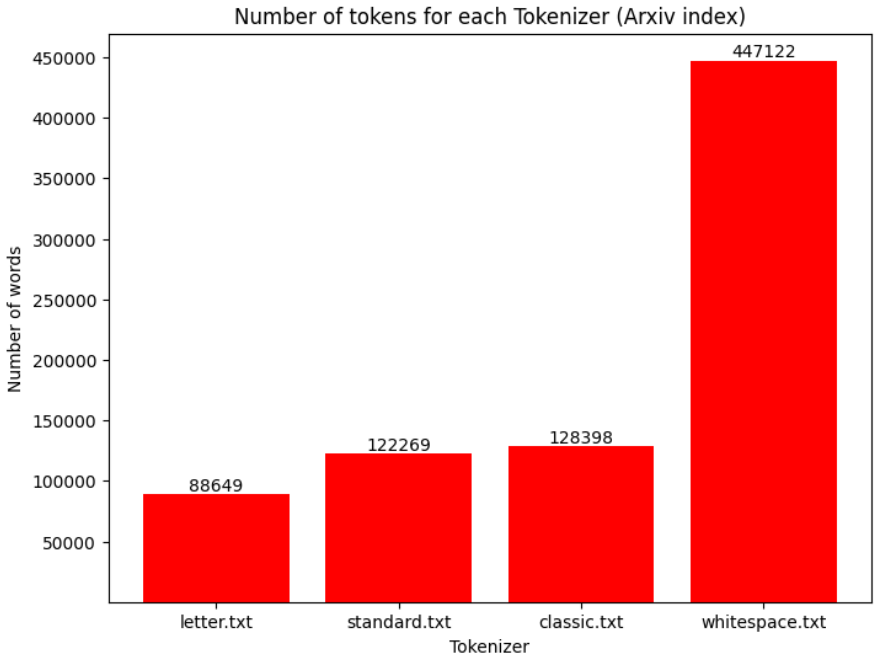
Prof: Ignasi Gómez Sebastià

24/9/2023 - 2023/24 Q1

1 Preprocessat en ElasticSearch

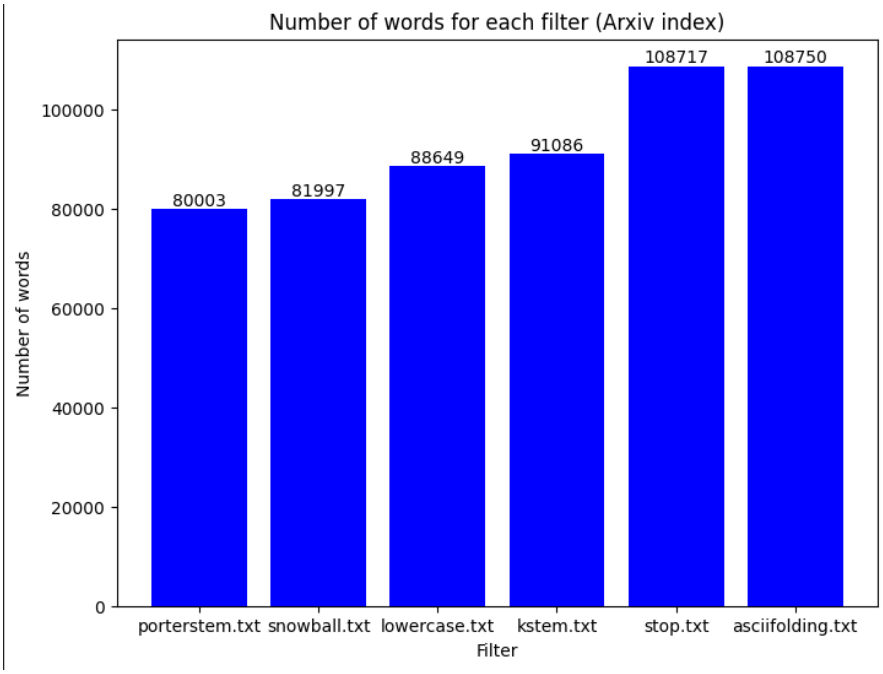
En aquest petit experiment posem com a objectiu, escollir diverses opcions de tractament de text que ofereix ElasticSearch (Diversos algorismes de tokenització, filtres, stemming...) per a realitzar un tractament dels texts més senzill i precís de les dades. Per aquest propòsit durem a terme una anàlisi qualitativa i quantitativa per a les diverses opcions, usant l'índex d'articles científics d'Arxiv com a dades.

Primerament, compararem diversos tokenitzadors, pel que fa al nombre de paraules diferents resultants del procés:



Com bé es pot apreciar, el tokenitzador més agressiu (similar a l'expresió regular que vaig crear en la pràctica passada per filtrar paraules) és el "letter" i llògicament el que menys és "whitespace".

Un cop comparats, fixarem per simplicitat el tokenitzador "letter", per a comparar els filtres entre si, de la mateixa manera que hem fet amb els tokenitzadors.



És bastant clar que els filtres d'stemming redueixen el nombre de paraules considerablement (entre 10% - 20%), però l'algorisme que ofereix millors resultats en aquest cas es tracta de l'algorisme de Porter.

També és notori mencionar que l'ordre dels filtres és important, ja que s'apliquen un darrere del següent i l'ordre llavors serà rellevant, ja que algunes paraules podrien ser filtrades quan són importants (Per exemple, un article de notícies que parli del "ME TOO", si apliquem primer lowercase i després stopwords s'esborraria, ja que "too" és una stopword anglesa, cal fixar-se que el contrari no es filtrarà).

D'entre els diversos tokenitzadors hem escollit usar "letter", ja que sent el més restrictiu ens lliurará de molts números i termes invàlids. Pel que fa els filtres, farem servir un algorisme de stemming, com "porterstem" mateix, també "lowercase", precedit de "stop". Aquesta configuració serà la que utilitzarem per a avaluar documents un cop implementat el model vectorial en l'apartat següent usant l'índex ARXIV.

2 Implementació TF-IDF

A continuació proposem una implementació de TF-IDF per a vectoritzar documents i permetre la cerca per similaritat entre aquests, que pot ser trobada al Jupyter Notebook adjunt o al módul TFIDFViewer.py. Aquest tractament dels documents, els converteix en vectors de mida arbitraria en un espai R^n, on component del vector representa un terme (t) del document(d) de l'índex D, amb un cert pes que és definit per la fórmula (On N indica nombre de documents i n_t el nombre de documents que conté t):

$$\text{TF-IDF}(t, d, D) = \left(\frac{f_{t,d}}{f_{\max,d}}\right) \times \log\left(\frac{N}{n_t}\right) \tag{1}$$

Aquesta representació dels documents és especialment útil, ja que podrem fàcilment comparar documents de mides diverses, mitjançant l'angle que formen, conegut com la similitud cosina. Si ens fixem en l'expressió per computar-lo, sabent que la funció rebrà vectors ja normalitzats, només caldrà fer un producte escalar:

$$\text{Cosine Similarity}(\mathbf{A}, \mathbf{B}) = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2}$$

Per accelerar l'algorisme podem aprofitar-nos del fet que ambdós vectors també ja estan ordenats alfabèticament pels seus termes. Llavors recorrerem els 2 vectors de la mateixa manera que faríem en un algorisme clàssic de fusió de llistes ordenades (fent ús de 2 punters (i,j) es recorren les dues llistes comprovant iteració per iteració si els elements actuals d'ambdues llistes són iguals o no, per a fer la intersecció d'elles. Acaba l'algorisme quan una de les dues llistes s'exhaureix), aconseguint una complexitat $O(min(n, m))$ on n,m són les respectives mides d'ambdós vectors, evitant una alta complexitat de l'algorisme "naive" $O(n + m)$.

Per més detalls d'implementació recomano donar un cop d'ull al script que es proveeix. La verificació del correcte funcionament ha estat feta revisant

el codi manualment minuciosament i l'execució de varies similituds entre documents de l'índex "docs" el qual ja hi han resultats a les transparències de l'assignatura de les similituds entre documents (Per exemple comprar els documents 3 i 4 ha donat 0.035).

3 Conclusions i dificultats tècniques

Primerament, cal comentar que la implementació de TF-IDF podria haver sigut més ràpida, per exemple, en la normalització, si els vectors entrats tinguessin un altre estructura, amb la qual es pogués fer ús de les funcions escalars de numpy, que són extremadament ràpides comparades amb la manipulació de llistes i tuples en Python. També cal veure que per com està feta la plantilla de la pràctica, no es pot pas comparar fitxers que estiguin buits, ja que es dona una excepció.

Concloem de l'experiment de filtres i tokenitzadors, que la paraula més popular de l'anglès és "the", però si filtrem stopwords pot ser "we", la qual no esperava sincerament.

Per a posar a prova el model TF-IDF hem pensat de fer diversos experiments amb l'índex d'articles científics. Primerament vam comparar els 2 primers fitxers de computer science updates

$$sim(000000, 000001) = 0.017 \tag{3}$$

Al principi vaig pensar que aquest resultat tant baix (1.7%) es devia a la diferència dels temes comentats (un parla d'escalabilitat i l'altre de models de big data), però després de fer una mitjana per a 20 parells diferents, en vaig obtenir 0.0135. Probablement, això és causat perquè tots els documents són curts abstracts, tots de temàtiques força diferents. De totes maneres els resultats tenen sentit si tenim en compte l'experiment següent: comparar 11 parelles de documents (computer science, astro physics), per trobar si aquesta mitja és similar a l'anterior. Sorprenentment 3 parells de fitxers van aconseguir una similaritat de 0.012 aproximadament i la resta aproximadament 0.005:

$$\frac{0.00338+0.01237+0.00499+0.00531+0.00650+0.00442+0.00680+0.00520+0.00415+0.01234+0.01243}{11} = 0.00708$$

Aleshores si comparem les dues mitges anteriors (Encara que una és més robusta que l'altre) ens podem adonar que efectivament és inferior, cosa que té sentit. Finalment, compararem els abstracts d'articles de física amb astrofísica, hauríem de notar certa similitud. El resultat és que si hi ha més similitud que en la comparació anterior (23% més), ja que hi haurà certs articles on els temes que es parlen són força semblants, però continua essent més baix que comparar dins d'una mateixa disciplina com hem fet abans amb computer science, el qual és una bona senyal.

$$\frac{0.01666+0.00164+0.00873+0.01643+0.00673+0.00469+0.00875+0.00281+0.01701+0.00486+0.01142}{11} = 0.0092$$

La comparació que hem fet ha estat més relativa que no pas absoluta, és com si la majoria dels documents no s'assemblessin entre si (Absolutament dona valors no molt grans, inferiors al 2% sempre), però això és degut a part de les temàtiques dels textos, el vocabulari emprat, els verbs usats, l'estil de l'escriptor i altres factors, que fan que els textos siguin diferents un d'altres (També hem de tenir en compte que hem tret stopwords, cosa que fa que perdin paraules en comú).