UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria
de Vilanova i la Geltrú

RECUPERACIÓ DE LA INFORMACIÓ (REIN)
Sessió 2 laboratori: Programes amb l'Elasticsearch

En aquesta sessió:

- Aprendrem com indicar a l'Elasticsearch que apliqui als documents diferents segmentadors (*tokenizers*) i filtres com l'eliminació de paraules funcionals (*stopwords*) i l'*stemming* de les paraules.
- Analitzarem com afecten aquests canvis als termes que l'Elasticsearch afegeix a l'índex i com afecten a les consultes.
- Completarem un programa per visualitzar els documents en el model vectorial amb pesos *tf-idf*.
- Calcularem la similitud entre documents fent servir la mesura cosinus.

1 Canvis en la indexació de l'Elasticsearch

Una de les tasques de la sessió anterior va ser eliminar del vocabulari dels documents tots aquells termes (o cadenes de caràcters) que no eren pròpiament paraules. Aquesta és una tasca freqüent i, per això, aquest tipus de BD tenen processos estàndard que ajuden a filtrar i reduir els termes que no són útils per fer consultes.

El text dels documents, abans de ser indexat, pot ser sotmès a una *pipeline* formada per diferents processos que n'eliminen allò que no sigui útil per a una determinada aplicació.

El primer pas de la pipeline sol ser un procés que converteix el text pla en un conjunt de tokens. Per exemple, podem segmentar el text fent servir els blancs i els signes de puntuació, o fer servir un analitzador específic per un idioma que detecti les paraules d'aquell idioma, o un parser d'HMTL/XML, etc.

En aquest apartat del manual de l'Elasticsearch s'expliquen els diferents tipus de segmentadors disponibles.

Un cop tenim els tokens, podem normalitzar-los i filtrar aquells que no siguin útils. Per exemple, normalment, els tokens es passen a minúscules perquè totes les aparicions de la mateixa paraula es corresponguin amb el mateix token independentment de si està en majúscules o no. També hi ha paraules "buides" (no tenen contingut semàntic) i que no són útils per les consultes, com ara articles, preposicions i adverbis; cada idioma té la seva pròpia llista de paraules buides que s'anomenen *stopwords*. Un altre procés de normalització dels tokens que és específic de l'idioma és l'*stemming*. L'arrel o lexema d'una paraula correspon a la part comuna de la paraula a partir de la qual es formen les seves variants per flexió o

derivació (afegint prefixos o sufixos). Per exemple, les paraules *paperassa*, *papereria* i *empaperar*, deriven totes de *paper*. La idea és que totes les variacions d'una paraula es representin amb el mateix token.

En aquest apartat del manual de l'Elasticsearch hi trobareu diferents possibilitats (exploreu les opcions sota el desplegable *Token filter reference*).

2 Recàrrega de l'índex

La primera tasca d'aquesta sessió és analitzar l'efecte que produeixen els canvis que fem en la pipeline sobre els tokens i la seva quantitat total. Teniu una nova versió de l'script d'indexació de la sessió anterior que s'anomena `IndexFilesPreprocess.py`. Aquest script té dos arguments addicionals `--token` i `--filter`.

L'argument `--token` canvia el segmentador de text; teniu quatre opcions: `whitespace`, `classic`, `standard` i `letter`. Useu cadascun d'ells amb els documents de les novel·les i compareu-ne els resultats. No canvieu el filtre que es fa servir per defecte (en aquest cas, només passar a minúscules). Mireu-vos la documentació per entendre què fan aquests segmentadors. Useu l'script `CountWords.py` de la sessió anterior per veure quants tokens s'obtenen.

Després d'això, useu el segmentador més agressiu i useu els filtres disponibles en l'script: `lowercase` (obvi), `asciifolding` (elimina caràcters no ASCII que usen molts idiomes), `stop` (elimina stopwords estàndard de l'anglès) i diferents algorismes d'stemming per l'anglès (`snowball`, `porter_stem` i `kstem`). Heu de fer servir l'opció `--filter`, que ha de ser l'última, i podeu posar diferents filtres separats per espais en blanc. Per exemple:

```
$ python3 IndexFilesPreprocess.py --index inovels --path /path/to/novels \
--token letter --filter lowercase asciifolding
```

Quina és la paraula més freqüent de l'anglès? Obtens els mateixos resultats sobre els diferents corpus (`20_newsgroups` i `arxiv_abs`)?

Com a extra, podeu aprendre com configurar l'analitzador de textos d'un índex i canviar l'script per fer servir encara més opcions.

Com a projecte addicional, podeu comprovar si tot aquest preprocessament canvia o millora l'ajust de la llei de Zipf.

3 Càlcul dels pesos *tf-idf* i de la mesura cosinus

L'objectiu d'aquesta part de la sessió és entendre l'esquema de pesos *tf-idf* per representar els documents com a vectors i la mesura de similitud cosinus. Completarem un script que

rep la ruta on es troben els fitxers, obté els seus identificadors (`id`) de l'índex, calcula els vectors *tf-idf* corresponents als documents, opcionalment mostra aquests vectors, i finalment calcula la seva similitud fent servir la mesura cosinus.

L'script `TFIDFViewer.py` conté un conjunt de funcions, incompletes, per fer tot això:

- El `main` segueix l'esquema que acabem de veure.
- La funció `search_file_by_path` retorna l'`id` d'un document de l'índex (la ruta ha de ser el path sencer on es trobava el document en el moment de la indexació, no només el nom d'un fitxer).
- La funció `document_term_vector` retorna dues llistes de tuples. La primera llista és de tuples (terme, freqüència del terme en el document), i la segona de tuples (terme, freqüència del terme en l'índex). Les dues llistes estan ordenades alfabèticament per terme.
- La funció (incompleta) `toTFIDF` retorna una llista de tuples (terme, pes) que representa el document amb l'`id` donat. Fa el següent:
  - Primer obté dues llistes amb les freqüències dels termes en el document i les freqüències dels termes en l'índex.
  - Obté el nombre de documents en l'índex.
  - Finalment, crea cadascuna de les entrades del vector, tupla (terme, TFIDF), que retornarà.

  La vostra tasca consisteix en completar els càlculs del valor *tf-idf* per omplir el vector. Només heu d'aplicar les fórmules que hem vist a classe.
- La funció (incompleta) `normalize` hauria de calcular la norma del vector (arrel quadrada de la suma dels components al quadrat) i dividir tot el vector per aquesta norma, de manera que el vector resultant tingui norma (longitud) 1. Completeu aquesta funció.
- La funció (incompleta) `print_term_weight_vector` escriu una línia per cada entrada del vector amb el format (terme, pes). Completeu aquesta funció.
- La funció (incompleta) `cosine_similarity` es pot implementar normalitzant primer els dos arguments i després calcular el seu producte escalar. Completeu aquesta funció. IMPORTANT: Ha de ser una implementació eficient de manera que, com a molt, faci un recorregut del vector. Feu servir la propietat que els vectors estan ordenats alfabèticament per terme.

Per calcular l'arrel quadrada i el log_{10}, podeu fer servir les funcions `sqrt` i `log10` de la biblioteca `numpy`. La biblioteca ja està importada en l'script com a `np`.

Per poder provar la vostra implementació, disposeu d'un conjunt de documents en el directori `docs` que es corresponen a l'exemple vist a classe en el Tema 2 Models de recuperació de la informació.

4 Exercicis

Quan ja tingueu el programa, proveu-lo amb les col·leccions de les sessions anteriors. Primer, proveu el vostre codi per calcular la similitud d'un fitxer amb ell mateix (què hauria de donar?).

També podeu crear-vos una col·lecció molt simple amb només dos documents i tres o quatre termes perquè pugueu comparar el resultat de l'execució amb els vostres càlculs fets a mà.

Podeu fer tot tipus de proves, per exemple, comprovar si els documents d'un subconjunt del corpus `20_newgroups` són més similars entre ells que amb els d'un altre subconjunt no relacionat (p.e., `alt.atheism` vs `sci.space`)?

5 Lliuraments

A la Tasca d'Atenea prevista per aquesta activitat, pengeu un breu report en format PDF (3-4 pàgines màxim). Recordeu d'incloure els vostres noms i el de l'activitat.

Per la primera part de la sessió, comenteu els efectes que heu observat en l'índex (mida, nombre de termes, etc). L'important és que escriviu les vostres conclusions, no que doneu xifres, taules o captures de pantalla.

Per la segona part de la sessió, expliqueu:

- Si heu entès què fa el programa i el que dóna, i quines proves heu fet (documents utilitzats).
- Si us ha funcionat fàcilment o heu tingut alguns problemes en la implementació. Si n'heu tingut, expliqueu breument quins.
- Qualsevol observació sobre les vostres proves i/o el que n'heu après.

Amb el report, també heu de lliurar els scripts Python que hàgiu modificat. Marqueu de forma visible, amb comentaris, les parts que heu canviat. Empaqueteu-ho tot en un fitxer .zip i pengeu-lo.