



RECUPERACIÓ DE LA INFORMACIÓ (REIN)

Sessió 1 laboratori: Elasticsearch i les lleis de Zipf i Heaps

En aquesta sessió:

- Aprendrem a usar Elasticsearch.
- Aprendrem a indexar un conjunt de documents de text i farem consultes simples en els documents indexats.
- Comprovarem si una col·lecció de textos satisfà les lleis de Zipf i Heaps.

1 Execució de l'ElasticSearch

[ElasticSearch](#) pot ser usat com a base de dades (documents) NoSQL amb la capacitat d'indexar i consultar documents de text. La base de dades funciona com un servei web en una màquina i s'hi pot accedir usant una API REST.

Per executar l'ElasticSearch en el vostre ordinador, us n'haureu de descarregar la darrera versió a:

<https://www.elastic.co/es/downloads/elasticsearch>

Trieu la versió que correspongui al vostre sistema operatiu. L'opció més senzilla és que us descarregueu les versions `.zip/.tar.gz`. També hi ha paquets estàndard per distribucions linux (`.deb/.rpm`) però implica executar l'ElasticSearch com un servei en el vostre ordinador. Si preferiu aquesta opció, podeu seguir les instruccions d'aquesta pàgina:

<https://www.elastic.co/guide/en/elasticsearch/reference/current/install-elasticsearch.html>

Una vegada descomprimit el fitxer que heu descarregat, disposareu d'un directori amb l'ElasticSearch. Si us moveu a aquest directori, només haureu d'executar-lo des d'un terminal:

```
$ bin/elasticsearch
```

Si voleu canviar els directoris per defecte on es guarden les dades i els logs, haureu d'editar el fitxer `elasticsearch.yml` que es troba en el directori `config` i canviar `path.data` i `path.logs` perquè apuntin als directoris on vulgueu que es guardin.

Per comprovar si l'ElasticSearch està funcionant correctament, teniu l'script `elastic_test.py` que podeu executar des de la línia de comandes fent:

```
$ python elastic_test.py
```



Si l'ElasticSearch està funcionant, veureu la resposta del servidor; altrament, veureu un missatge indicant que no està executant-se.

2 Indexació i consulta

Com hem comentat anteriorment, l'accés a ElasticSearch es fa mitjançant una API REST. Això vol dir que podem operar amb la BD utilitzant el protocol HTTP (com qualsevol altre servidor web). L'ElasticSearch defineix un conjunt de mètodes que corresponen a totes les accions disponibles. Podeu accedir a tota la documentació seguint aquest [enllaç](#).

Per simplificar la tasca, usarem dues llibreries de Python ([elasticsearch](#) i [elasticsearch-dsl](#)) per accedir a l'ElasticSearch. La primera és més general però oculta menys la complexitat de les crides a l'API. La segona està més dirigida a les capacitats de cerca i és més fàcil d'usar per fer les consultes a l'ElasticSearch. En els scripts que teniu disponibles s'usen les dues llibreries depenent de les operacions que s'hagin de fer.

2.1 Indexació

Si ho comparem amb les bases de dades relacionals, un índex d'ElasticSearch correspon a una taula. No hi ha un esquema específic (conjunt de columnes) associat amb l'índex; s'hi poden inserir diferents tipus de documents que poden tenir camps diferents. Se li pot indicar a la BD quins camps es volen indexar. Tots els documents s'envien a la BD serialitzats usant el format JSON; així, un document presenta la forma següent:

```
{"field1": "2020-09-23",  
  "field2": "Some text here",  
  "field3": 33  
}
```

Descarregueu-vos els següents arxius comprimits: [novels](#), [20newsgroups](#) i [ArXiv abstracts](#). Descomprimiu-los en el vostre directori de treball (o a /tmp si no disposeu de prou espai a disc), i anomeneu als directoris `novels`, `20_newsgroups` i `arxiv_abs`. Aquests arxius contenen, respectivament:

- Unes quantes novel·les a l'atzar i altres textos en anglès, extrets del [projecte Gutenberg](#); la majoria d'elles són de finals del segle XIX principis del segle XX.
- Textos de 20 grups de discussió (*newsgroups*) d'Usenet sobre diversos temes; és un corpus clàssic per l'avaluació en RI (podeu obtenir més info [aquí](#)).
- Una petita col·lecció de resum d'articles recents de diferents àrees extreta d'[aquí](#).



Dins dels fitxers de la sessió hi ha un script anomenat `IndexFiles.py`. Obriu l'script en un editor de textos.

L'script conté bàsicament dues parts. La primera llegeix tots els documents recorrent de forma recursiva la ruta que rebí com a paràmetre i crea una llista d'operacions de l'ElasticSearch. L'API té un mètode per indexar un únic fitxer però, atès que nosaltres volem indexar-ne molts, usarem el mètode massiu (**bulk**) que permet executar diverses crides d'ElasticSearch com una de sola.

Una operació **bulk** és un document especial que indica quin tipus d'operació s'ha d'executar (`_op_type`), en aquest cas **index**, l'índex en el que es farà l'operació (`_index`) i la informació del document: el tipus de document i els camps del document. En aquest cas, hem usat com a tipus **document** i incloem dos camps del document: **path** per la ruta del fitxer i **text** pel contingut del fitxer.

La segona part de l'script fa les operacions amb l'ElasticSearch. El primer que necessitem és un objecte (instància) ElasticSearch que gestioni la connexió amb la BD. Si tenim el servei en la màquina local amb la configuració per defecte, no necessitem més paràmetres; sinó, hem de configurar l'objecte adequadament. Amb aquest objecte crearem l'índex pels documents (en cas d'existir, s'esborraria) i es cridarà el mètode massiu per fer totes les insercions.

Per usar l'script amb els documents que us heu descarregat, podeu fer, per exemple:

```
$ python IndexFiles.py --index inovels --path /path/to/novels
```

que inserirà tots els documents de la col·lecció **novels** en l'índex **inovels**.

2.2 Consulta

Després d'inserir tots els documents a l'índex ja podem fer consultes a l'ElasticSearch usant termes específics o consultes més complicades. L'ElasticSearch té un DSL (*Domain Specific Language*) per especificar què volem buscar. Sense entrar en la complexitat de tot el que es pot fer, només indicar que aquest tipus de BD permet més flexibilitat que les bases de dades relacionals clàssiques. A més de concordances (*mappings*) exactes, podem fer *mappings* difusos (*fuzzy*), obtenir suggerències, ressaltar les parts del documents que inclouen els termes de cerca... Bàsicament, tot allò que podem fer amb un cercador web perquè aquest tipus de BD és el mateix que el es troba darrera d'una consulta a Google, per exemple.

L'script `SearchIndex.py` permet fer consultes en un índex en la nostra BD d'ElasticSearch. Té tres arguments: `--index`, `--text` i `--query`.

L'argument `--text` és més prioritari que `--query` (no podem usar els dos alhora) i permet buscar una paraula en el camp **text** dels nostres documents. El que retorna són tots els



mappings exactes amb l'identificador del document (*id*), la ruta del fitxer que fa *matching* i els 10 primers caràcters del document.

L'argument `--query` permet usar la sintaxi de LUCENE¹ per fer les consultes. Aquesta sintaxi ofereix operacions booleanes com AND, OR, NOT (sempre amb majúscules) o consultes difuses usant `~n` on *n* indica quantes lletres de la paraula poden incompatibilitzar per considerar-la una coincidència. Usant aquest argument, la consulta retorna l'id dels documents, la ruta i els 10 primers caràcters del document.

Obriu l'script en un editor i mireu la part del codi on es construeix la consulta. Per entendre'l millor, consulteu la documentació de `elasticsearchdsl`.

Ja podeu usar l'script i aneu fent proves. Per exemple:

```
$ python SearchIndex.py --index inovels --text good
$ python SearchIndex.py --index inovels --query good AND evil
$ python SearchIndex.py --index inovels --text angle
$ python SearchIndex.py --index inovels --query angle~2
```

Cada consulta torna el nombre de documents coincidents. Comproveu com canvia aquest nombre en augmentar els valors de *n* en les consultes difuses.

3 Les lleis de Zipf i Heaps

Anem a analitzar la distribució de les paraules en aquests texts. Concretament, si segueixen les lleis de Zipf i Heaps. Per fer això, usarem el corpus de novel·les (*novels*), però és interessant veure com es comporten en la resta de corpora.

L'script `CountWords.py`, que teniu en els fitxers d'aquesta sessió, llegeix l'índex especificat i escriu en el canal estàndard de sortida tots els termes que conté amb les seves freqüències. L'script té dos arguments: `--index` i el flag `--alpha` que permet retornar la llista de paraules ordenada alfabèticament en lloc d'ordenada de forma ascendent per freqüència. Executeu-lo redireccionant la sortida a un fitxer. Fixeu-vos que la darrera línia conté el nombre de paraules diferents i el nombre total de paraules de l'índex.

Obriu aquest fitxer, ordeneu les paraules alfabèticament i elimineu els termes que penseu que poden distorsionar les estadístiques (nombres, URL, nombres binaris o dades il·legibles, dates...).

Per la llei de Zipf, comproveu si la distribució de rang-freqüència té forma de *power law*. Quins són els paràmetres ($f = c * (rang)^{-\alpha}$) que descriuen millor el que veieu? Podeu usar

¹LUCENE és el sistema d'indexació de codi obert que usen aquest tipus de BD.



un full de càlcul i provar diferents parelles (α, c) per intentar imitar el nombre d'ocurrències d'una paraula en funció del seu rang. Dibuixar una gràfica (amb escala lineal o bé log-log) us pot ajudar. També podeu usar les llibreries de Python tant per fer plots (`matplotlib`, `seaborn`) com per ajustar funcions (`numpy`, `scipy`). Els enllaços següents us poden ajudar:

- [Breu tutorial de matplotlib](#)
- [Breu tutorial de seaborn](#)
- [Ajust de funcions amb scipy](#)

No feu massa cas dels primers termes (els més freqüents) perquè generen soroll. El que importa és la llarga cua. Amb la vostra fórmula, obteniu de forma aproximada les freqüències 1, 2, 3... en el mateix lloc que ocupen en les dades?

Nota: Fer un plot de les dades per comprovar que s'obté una corba decreixent que tendeix a zero, no és suficient. Segur que obtindreu una corba així. L'important és si la corba és realment una power law com prediu la llei de Zipf (o és una exponencial?, o una altra funció amb una ràtio de decreixement diferent?).

Per la llei de Heaps, proveu si el nombre de termes diferents en un tros de text d' N paraules és de l'ordre de $k * N^\beta$ per alguna k i β . Per fer això:

1. Creeu índexs que continguin un nombre diferent de termes (diferents novel·les o trossos de novel·les).
2. Useu el programa per comptar el nombre de paraules de cada índex i el nombre de paraules diferents, per cadascun d'ells.
3. Proveu de trobar k i β que expliquin aquests resultats.

4 Lliuraments

Heu d'escriure un breu report en format PDF (2-3 pàgines màxim), on expliqueu els resultats que heu obtingut, com els heu obtingut i les conclusions que n'heu tret, en les proves de les lleis de Zipf i Heaps.

Per la llei de Zipf, heu de donar els valors dels paràmetres estimats per descriure la distribució de rang-freqüència de les dades. Després, compareu els valors predits amb els paràmetres estimats, per a determinades freqüències.

Per la llei de Heaps, cal que doneu els valors dels paràmetres estimats per descriure el nombre de termes diferents. Després, compareu els valors predits amb els paràmetres estimats, per a diferents quantitats de documents (els nous documents, s'afegeixen als inicials, no els substitueixen).



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

**Escola Politècnica Superior d'Enginyeria
de Vilanova i la Geltrú**

El lliurament d'aquesta activitat es farà a través d'Atenea en la Tasca prevista per fer-ho i abans de la data límit.