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

# RECUPERACIÓ DE LA INFORMACIÓ (REIN)
## Sessió 1 de 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'emmagatzemar, 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.

El binari de l'Elasticsearch es troba a `/usr/share/elasticsearch/bin`

Per executar l'Elasticsearch en el vostre ordinador, us podeu descarregar aquesta mateixa versió des de:

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

A partir d'aquest punt, depenent de com hàgiu fet la descàrrega (`.zip/.tar.gz` o `.deb/.rpm`, per exemple), hauríeu de seguir els passos d'instal·lació i execució seguint els enllaços corresponents en la mateixa pàgina.

També s'explica com canviar els directoris per defecte on es guarden les dades i els logs, editant el fitxer `elasticsearch.yml` i modificant `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:

```
$ python3 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 biblioteques 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'utilitzar 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": "2023-09-13",
 "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 resums 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 rebi 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, farem servir 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 què es farà l'operació (`_index`) i la informació del document (camps del document). En aquest cas, hem inclòs 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'index pels documents (en cas d'existir, s'esborraria) i es cridarà el mètode massiu per fer totes les insercions.

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

```
$ python3 IndexFiles.py --index news --path /path/to/20_newsgroups
```

que inserirà tots els documents de la col·lecció `20_newsgroups` en l'índex `news`.

## 2.2 Consulta

Després d'inserir tots els documents a l'índex ja podem fer consultes a l'Elasticsearch fent servir 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 que es troba darrera d'una consulta a Google, per exemple.

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

L'argument `--text` és més prioritari que `--query` (no podem fer-los servir tots dos alhora) i permet buscar una paraula en el camp `text` dels nostres documents. El que retorna són tots els *matchings* exactes amb l'identificador del document (`id`), la ruta del fitxer que fa *matching* i les parts destacades que contenen la paraula.

L'argument `--query` permet usar la sintaxi de LUCENE[1] per fer les consultes. Aquesta

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

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 ser incompatibles 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 `elasticsearch-dsl`.

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

```
$ python3 SearchIndex.py --index news --text good
```

```
$ python3 SearchIndex.py --index news --query good AND evil
```

```
$ python3 SearchIndex.py --index news --text angle
```

```
$ python3 SearchIndex.py --index news --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, farem servir el corpus de novel·les (`novels`), però és interessant veure com es comporten en la resta de corpus.

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 l'opció `--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 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 ($f = \frac{c}{rang^{\alpha}}$) que descriuen millor el que veieu? Podeu usar un full de càlcul i provar diferents parelles ($\alpha$, $c$) 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 biblioteques 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, comproveu si el nombre de termes diferents en un tros de text d'N paraules és de l'ordre de $k \times N^{\beta}$ per algunes $k$ i $\beta$. En aquest cas, és millor fer servir només el corpus de novel·les perquè els documents són més llargs. Per fer això:

1. Creeu índexs que continguin un nombre diferent de novel·les o, més concretament, de trossos de novel·les perquè n'hi ha de mides molt diferents.
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 $\beta$ que expliquin aquests resultats.

# 4 Lliuraments

Heu d'escriure un breu report en format PDF (3-4 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 millors valors que hàgiu trobat dels paràmetres estimats per descriure la distribució de rang-freqüència de les dades ($\alpha$, $c$). Heu d'explicar com els heu trobat i quin mètode heu fet servir per trobar-los. Compareu els valors predits amb els paràmetres estimats, per a determinades freqüències, amb els valors reals i expliqueu les conclusions que en traieu.

Per la llei de Heaps, cal que doneu els valors dels paràmetres estimats per descriure el nombre de termes diferents ($k$, $\beta$). 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), amb els valors reals.

Si feu proves de les lleis de Zipf i Heaps amb corpus diferents (novel·les, grups de discussió o articles), observeu diferències en els resultats?

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