Lab Session 2: Intro to ElasticSearch

Alex Carrillo Alza, Roger Creus Castanyer
Cerca i Anàlisi de la Informació, GCED, UPC

16 September 2019

1 Running ElasticSearch

In our case, we try to use ElasticSearch in our personal computer by installing it on macOS with the *Homebrew* package manager. To install with Homebrew, you first need to tap the Elastic Homebrew repository:

```
$ brew tap elastic/tap
```

Once you have tapped the Elastic Homebrew repo, you can use `brew install` to install the default distribution of ElasticSearch:

```
$ brew install elastic/tap/elasticsearch-full
```

As we have disk space, we can run directly the script that starts the database from `/usr/bin` and it is not necessary to create a `/tmp` directory. However, to be able to access the files of this lab session through a symlink we must work in `/usr/local/bin` directory in order to prevent changes to several core parts of the OS:

```
$ mkdir -p /usr/local/bin
$ ln -s /usr/bin/elasticsearch /usr/local/bin/elasticsearch
```

After running ElasticSearch we can test it is working with the provided `elastic_test.py` script and throwing the URL `localhost:9200`.

2 Indexing and querying

2.1 Anatomy of an indexing

In order to index documents sent to the DB, we used the script named `IndexFiles.py` which traverses recursively a set of documents, sets up its information and indexes it to the ElasticSearch object. Doing so, we get the following:

- Index `news` with 20089 files from `/20_newsgroups`
- Index `novl` with 33 files from `/novels`
- Index `arxv` with 53001 files from `/arxiv`

2.2 Looking for mr goodword

With the help of the `SearchIndex.py` script, we play a little bit and get the following results. E.g. for the word `good` in index `news` get 3813 documents and for `angle` 92. For the queries words `good AND evil` we obtain 146 documents.

Now, for fuzzy searches things get tricky. When using `~n` with n indicating how many letters of the word can mismatch in the search, the number of documents found rapidly increased as n was higher (as we were expecting). However, it should be noted that using an n greater than 3 did not return changes in the query at all. Later, we discovered this fact was given due to a default value of $max\_expansions = 2$ set up for efficiency reasons by ElasticSearch. I.e. the query uses $n = min(n, max\_expansions)$. An example of this behaviour is, given the query of the word `behave` in `news` with $n = \{0, 1, 2, 3\}$, the return of $\{67, 98, 10000, 10000\}$ documents, respectively.

3 Redoing Session 1

After having worked with queries and LUCENE syntax, we can compare the results obtained in lab Session 1 (when extracting words from `/novels` text) with the ones given by ElasticSearch. As we check the number of words indexed, we note that our function collects 52134 words, a bit less than those 61825 words obtained with ElasticSearch. This is given by the fact that our implementation did not properly process hyphenated words (e.g. the word *part-time* was left as *part*), so it missed some words. However, if we compare the corpus of words, both lists contain quite similar frequencies and only some words precede another in ranking.

4 Conclusions

All in all, the fact of having set up ElasticSearch in our own laptops and having tested a sheer volume of documents for indexing and querying has given us a broader perspective on how a NoSQL DB works and a deeper look in scopes out of the focus of this report.