Information Retrieval and Analysis
OCTOBER 2020

Lab Session 2: Intro to ElasticSearch

PROJECT REPORT

Elías Abad Rocamora
Victor Novelle Moriano
Barcelona, UPC - FIB & FME

Introduction to ElasticSearch and installation

The first step we had to perform in this laboratory session was to understand the functioning of *ElasticSearch*, the NoSQL database that would be used in this project to perform all the requested operations. In order to understand its structure and how the data is managed, we consulted the auxiliary link provided in the documentation. The comprehension of this tool was quite simple, as well as the installation and running, which we performed successfully and without any problems following the *Running ElasticSearch* section provided on the lab documentation.

Indexing

After understanding how *ElasticSearch* works, we proceeded to understand the codes provided. The first code, *IndexFiles.py*, creates an *ElasticSearch* index with fields 'path' and 'text' for every document in a given path. To facilitate the connection with the *ElasticSearch* server, the client is only called once and given all the operations via the *bulk* command, instead of calling it for each document we want to index.

Querying

After we had all of the documents indexed, we proceeded to apply queries to *ElasticSearch* to find the desired terms. To perform these queries, we used the *SearchIndex.py* code provided.

The first step was analyzing the code to understand how the *text* and *query* parameters worked (*query* allows the use of logic operators and a fuzzy search). In this analysis, we found that some modifications could be applied to the code conserving its functionality. For example, in the *text* query, a *multi_match* operation was applied, but only one field was provided, being equivalent to apply a *match* operation. This "finding equivalences" activity helped us to further understand the code and the functionalities that *ElasticSearch* provides.

The results we obtained in this section are:

- **--text good:** 3813 documents.
- **--query good AND evil:** 146 documents.
- **--text angle:** 92 documents.
- **--query angle~2:** 1535 documents.

Counting words

When comparing the results given by *CountWords.py* and the ones given by the code implemented in *Lab 1*, we notice that there are some differences in the frequencies given by both algorithms. The frequency given by our code appears to be always larger than the one given by *CountWords.py.* For example, our code encounters 206665 times the word "the", while *CountWords.py*, returns 206546. We don't know why this happens, but the difference is not significant and the order in which the words are placed in the list seems to be right.