UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

CAIM Lab, Session 2:

Programming with ElasticSearch

Fall 2021-22

Marco Madalin Farcas Vinter
Gerard Caravaca Ibañez

Index

1.The index reloaded

In this lab section our goal is to test different index tokens and filters in order to analyze how it affects the total number of words in the index and also which ones are affected by. The first step was using the different tokens in order to find the most aggressive of them, the main changes that we noticed with each one using the default lowercase asciifolding filter were:

- **Whitespace**: Very radical, it separates words by whitespaces as the name indicates and also leaves dots, parentheses, numbers.
- **Classic:** It removes dates and punctuations
- **Standard:** Quite similar to classic but also deletes numbers and underscores
- **Letter:** No unusual behaviour, although this was the most aggressive filter.

Now using letter we tested the remaining filter stop (the default was lowercase asciifolding), and also the snowball stemming as seen below (Fig 1.1). We expected stop to perform the most aggressive as it removes all english stopwords, and as expected we can see that it behaves as so, although it came as a surprise that many subjects were not deleted, this was due to elasticsearch's register of stopwords.

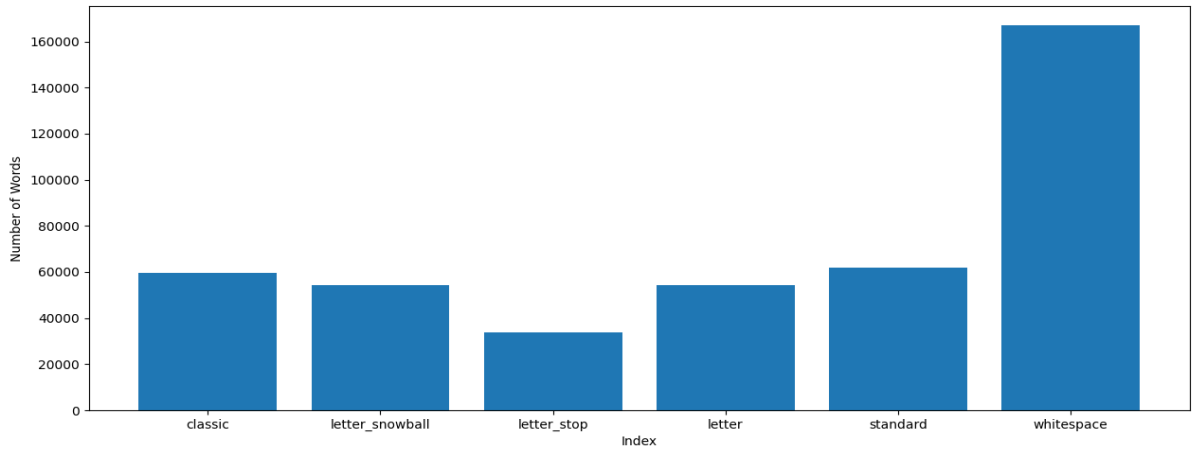

Fig 1.1: Word count with each token

After this experiment our most common words in the mix of news and novel index corpus is the letter **i** with **39994** repetitions, this is due to the stop filter removing stopwords; if we look just in the letter lowercase asciifolding we see that the most common word is, as expected, **the** with **206706** repetitions.

Now for the rest of this lab assignment our default settings are token:letter, and filter: lowercase asciifolding stop with the news-novel index:

--index newsnovels5 --path ./novels --token letter --filter lowercase asciifolding stop

2.Computing tf-idf ’s and cosine similarity

This second section consists of seeing what similarity there is between two documents using, as the name of this section implies, the tf-idf scheme for representing documents and cosine similarity measure. For this we have to complete the code provided to us with the formulas seen in theory class; the implementation did not pose much of a challenge besides thinking how to make the cosine_similarity function efficient, although it was pretty clear that as the vectors were sorted we could use the merge function from the merge sort algorithm with a linear cost O(n) by just taking the minimum of both vectors, the rest were just applying the formulas directly or basic printing.

In order to check if our implementation was correct we tried a test case scenario from the theory slides seen in class, the test was performed with the following commands:

1. *python3 IndexFilesPreprocess.py --index docs1 --path ./docs --token letter --filter lowercase asciifolding stop*
2. *python3 TFIDFViewer.py --index newsnovels5 --files ./docs/3 ./docs/4 > proveSim.txt*

The file obtained (Fig 2.1) coincides with the one seen in the slides, so we can ensure our code is correct, the following i:

TFIDF FILE ./docs/3
('five', 1.8073549220576042)
('four', 0.4074641404454827)
('one', 0.4074641404454827)
('three', 0.07413080711214934)

TFIDF FILE ./docs/4
('one', 0.305598105334112)
('six', 0.611196210668224)
('three', 0.055598105334112004)
('two', 1.8073549220576042)

Similarity = 0.03505

Fig 2.1: Test case file content

3.Experimentation

Now as a final section we shall see some experiments with different types of documents in order to check their similarity.

3.1.Similarity between two novels from the same author

DickensAChristmasCarol
DickensGreatExpectations

Similarity = 0.01715
DickensAChristmasCarol
DickensThePickwickPapers
Similarity = 0.01280
DickensGreatExpectations
DickensThePickwickPapers
Similarity = 0.03933

3.2.Similarity between different type of texts

The following dump is the cosine similarity from the experimentation performed, a much clearer sight can be seen in the following page's plot (Fig 3.1).

allNovels = 0.02363

onlyDickensNovels = 0.02309

math = 0.03491

religion = 0.02315

mixed = 0.05162

1mathvs1novel = 0.02133

1mathvs1religion = 0.00236

1religionvs1novel = 0.10495

Some interesting observations we noticed were:

- The comparison between Dickens novels is just as small as that of all the novels put together.
- The similitude between mixed texts is greater than that of texts on the same topic
- It is clear that the maximal difference is between religion and math texts. It may be because they are very different topics and not very long texts.
- Surprisingly, the similarity between a religious text and a novel is higher compared to the other cases comparing two different types.

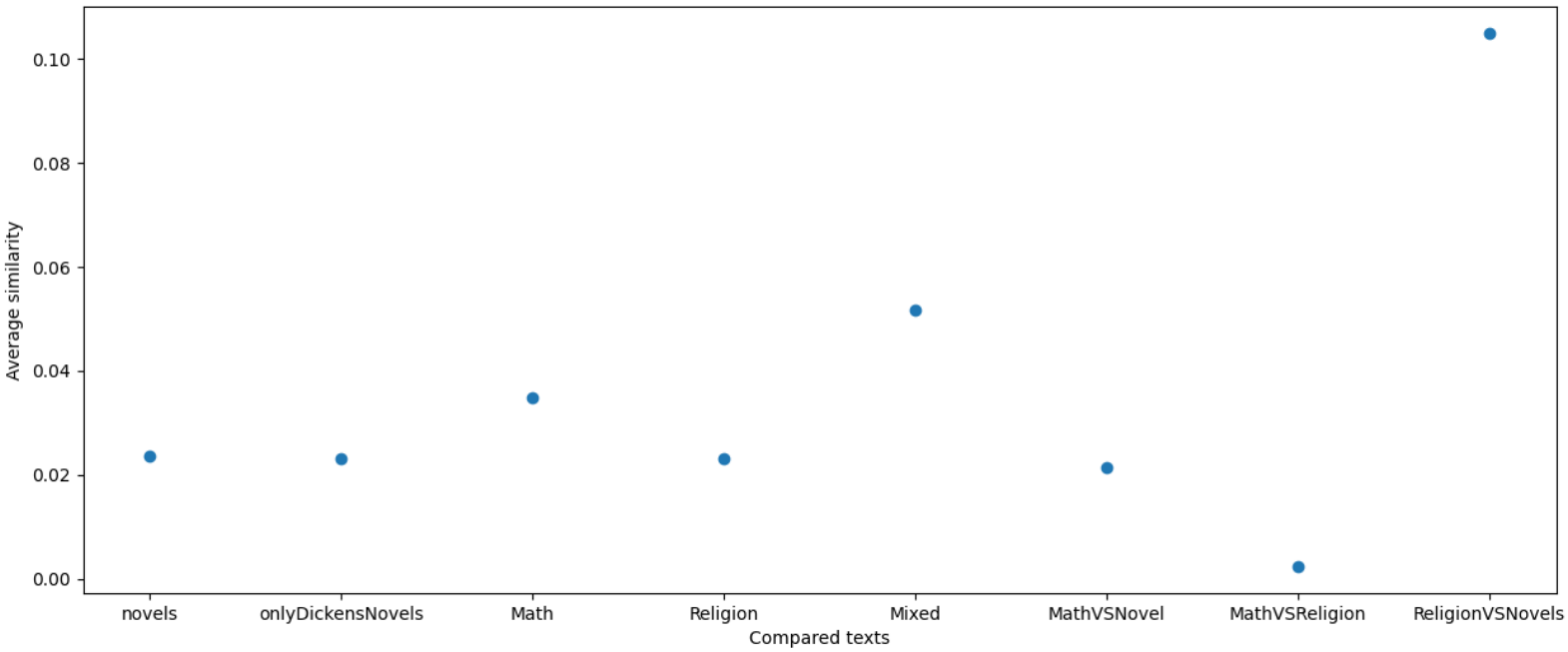

Fig 3.1: Similarity between different type of texts