TDT4117 Information Retrieval - Autumn 2022
Assignment 4
Deadline for delivery is 3.11.2022

October 2022

Important notes

Please carefully read the following notes and consider them for the assignment delivery. Submissions that do not fulfill these requirements will not be assessed and should be submitted again.

1. The assignment must be delivered in **ZIP format** containing code, report as pdf and other complementary content. Other formats such as .docx and .txt are not allowed.
2. The assignment must be **typed**. Handwritten assignments are not accepted.
3. Please be precise, to the point, and use figures/examples when it makes sense.
4. You may work in groups of maximum 2 students.

Task 1: Text Indexing

Text: Intelligent behavior in people is a product of the mind. But the mind itself is more like what the human brain does.

a) Construct an inverted file/list for the text above. Hint: remove all stop-words first

b) Use the above text to construct an inverted list with block addressing. State any necessary assumption.

c) Construct a partial vocabulary suffix tree and tree of the above text.

d) Construct a simple inverted index by using a posting list for the following document collection. State any necessary assumption.

Documents:

```
D1: Although we know much more about the human brain than we did even
D2: ten years ago, the thinking it engages in remains pretty much a total
D3: mystery. It is like a big jigsaw puzzle where we can see many of the
D4: pieces, but cannot yet put them together. There is so much about us
D5: that we do not understand at all.
```

Step 1: Create wordlist for all documents without stop-words

Step 2: Construct posting list

Task 2 : Index Analysis Using Lucene

For the rest of this assignment, you will be dealing with text indexing using the Apache Lucene framework via Elasticsearch Stack. Lucene's general goal is to provide open-source search software. Elastic provides an easy to use HTTP-Wrapper for Lucene. The projects homepages can be found at: `http://lucene.apache.org/` and `https://www.elastic.co/elastic-stack/`.

a) Give a short explanation of the Elastic Stack (ELK) and Lucene

Look up what the ELK stack provides and what it provides. Further check out Lucene and its capabilities plus how it is utilized within the ELK stack. How to use the Lucene query language and the ELK stack query language (KQL).

b) Implement a simple text indexer

Write down your idea/planned solution of a simple text indexer before you implement it. After, implement the simple text indexer in Python. **It does not need to be perfect.**

c) Index the files found in the archive DataAssignment4.tar.gz available on Blackboard

1. Index the documents with your indexer.

2. Load the given documents into the ELK stack (docker-compose and base Jupyter notebook provided).

Requirements: Docker and docker-compose are needed for this assignment. Docker can be installed by following this guide: `https://docs.docker.com/desktop/` and docker-compose `https://docs.docker.com/compose/install/`.

This setup provides you with an instance of Elasticsearch, Kibana and Jupyter/Python.

- Start the environment: *docker-compose up*
- Elasticsearch API: `http://elasticsearch:9200` (inside a Jupyter notebook) or `http://localhost:9200` (outsite Jupyter)
- Kibana GUI: http://localhost:5601/
- Jupyter/Python: http://127.0.0.1:8888/ (get the access token via the logs: *docker logs FOLDERNAME-jupyter*)

d) Search queries

After indexing the documents, perform the following searches with your own implementation and with the ELK stack:

- claim
- claim*
- claims of duty

Report the matching document. Do the resulting documents always match between your implementation and the ELK search?

e) Dive into a search query

- How is the query *"claims of duty"* handled in your implementation and in the ELK stack?
- Is there a similar query that provides the same result in your implementation and in the ELK stack?
- Does your result from the previous step for this query produce the expected result? Reflect based on your assumptions in the beginning.

f) Index and search in more documents

The previous example only contains 6 small files. To demonstrate the advantage of ELK/Lucene's index, we ask you to use the *Enron Email Dataset*, which contains more than 500,000 e-mails (There is a prepared code snipped (Jupyter Notebook) to reduce the number of emails to 100,000). First, download and extract the following archive: `https://www.cs.cmu.edu/~enron/enron_mail_20150507.tar.gz`. Using the same procedure as in the previous example, index the emails (with ELK and optional with your implementation) and search for the following terms "Norwegian and University and Science and Technology" and "Norwegian University Science Technology" in any of the e-mails. Which of the emails actually is around NTNU?