Assignment 4 - Text Indexing using Lucene

TDT4117 - Information Retrieval (Autumn 2024)

DUE: **November 6, 2024 23:59**

Start Here

- **Collaboration policy:**
  - You are expected to comply with the University Policy on Academic Integrity and Plagiarism.
  - You are allowed to talk with other students on homework assignments.
  - You can share ideas but not solutions; you must submit your solutions individually or as a group of two.
  - All submissions will be compared against all works submitted this semester and in previous semesters.
- **Submissions:**
  - Note that if you submit your work multiple times, the most recent submission will be graded.
  - You can do this assignment individually or in a team of two people; if you are in a team, just a single submission from either of the team members on Blackboard is sufficient.
  - When you are done, save the PDF report and Notebook files in a single .zip file with the filename **'group_no_Assignment4_TDT4117.zip'** and upload the .zip file on BlackBoard via the submission link for the assignment. Also, include the names of team members in the Notebook (or only your name if you are working on the assignment individually).
  - The assignment must be delivered in .zip format. Other formats, such as .py and .ipynb, are not allowed.

  Important: *Post all questions on the Ed platform and only send an email in case of (rare) anomalies!*

Task 1: Text Indexing

Inverted list

Use the provided text prompt to answer questions (a) to (c).

Text: *Ocean environment is a deep and complex ecosystem filled with diverse creatures. Yet, much about the ocean lacks exploration.*

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

b) Create an inverted list with block addressing. State any necessary assumption.

c) Construct a partial vocabulary suffix tree and tree.

Posting list

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

- **D1:** The universe is vast and mysterious, filled with countless stars and galaxies.
- **D2:** Each star may host its own planets, some of which could harbor life.
- **D3:** Scientists explore these possibilities through advanced telescopes and technology.
- **D4:** Yet, many questions about our universe remain unanswered, driving curiosity and research.
- **D5:** The quest for knowledge continues, pushing the boundaries of human understanding.

Task 2 : Indexing using Lucene

In this task, you will index text using the Apache Lucene framework via the Elasticsearch Stack. Lucene's primary goal is to provide open-source search software, while Elastic offers an easy-to-use HTTP wrapper for Lucene. You can find the project homepages at:

- http://lucene.apache.org/
- https://www.elastic.co/elastic-stack/

a) **Explanation of the Elastic Stack (ELK) and Lucene:** Research the features provided by the ELK stack and understand the capabilities of Lucene. Examine how Lucene is utilized within the ELK stack and explore the Lucene Query Language (LQL) as well as the Kibana Query Language (KQL).

b) **Implement a Simple Text Indexer:** Implement a simple text indexer in Python. Note that it does not need to be perfect.

c) **Index Documents:** Index the 'Texts' files found in the archive `DataAssignment4.tar.gz` available on Blackboard. Perform the following steps:

   i. Index the documents using your custom indexer.

ii. Load the indexed documents into the ELK stack. We have provided a docker-compose file and a base Jupyter notebook for your convenience.

Requirements: To use the ELK stack, you need Docker and Docker Compose. Follow the link below for a guide on how to install Docker on your computer: https://docs.docker.com/desktop/ and Docker Compose at: https://docs.docker.com/compose/install/.

Setup Instructions: Follow the steps highlighted below to start instances of Elasticsearch, Kibana, and Jupyter/Python.

- To start this setup, navigate to the directory containing your docker-compose.yml file and run: `docker-compose up` or `docker compose up` for newer versions of Docker.
- Use these to access Elasticsearch API:
  - Inside a Jupyter notebook: http://elasticsearch:9200
  - Outside Jupyter: http://localhost:9200
- Kibana GUI: http://localhost:5601/
- Jupyter/Python: http://127.0.0.1:8888/ (Access token can be found in the logs: `docker logs <jupyter-container-name> (for this project try: docker-compose logs elasticsearch)`).

Once the environment is started, we recommend relaunching your Jupyter notebook. After that, upload the provided template notebooks into the Jupyter Notebook.

d) **Search Queries:** After indexing the documents, perform the following searches using your own implementation and the ELK stack:

- `money`
- `money *`
- `money for charity`

Report the matching documents. Are the resulting documents always the same in your implementation compared to the ELK search?

e) **Dive into a Search Query:**

i. Analyze how the query `money for charity` is handled in both your implementation and the ELK stack.

ii. Identify if there is a similar query that yields the same result in both implementations.

iii. Assess whether the results for the previous query produced the expected outcomes. Reflect on your initial assumptions.

Important

We recommend you check elastic.co documentation: https://www.elastic.co/guide/en/elasticsearch/client/python-api/current/index.html.