Batch Retrieval - A Search Engine

CSE272 HW1 Report

Spring 2021, May 9

Alex Salman - aalsalma@ucsc.edu

1. **Software:**
    - **My design decisions and high-level software architecture**
        - When I started working on this project, I wanted to store the queries in a data structure but I found out later on that it is not necessary and queries could be processed on the fly
        - I also decided to store the documents on a RAM directory by using PyLucene, a wrapper of Lucene for Python
        - The software architecture involves reading documents and queries files in the main.py file, indexing documents and removing some stop words in index.py file. After that in the search.py file, the queries processed, stop words removed, and search/score/rank happens
    - **Major data structures**
        - Abstract data structure (list) to store the lines of the queries and documents while processing them
    - **Programming tools and libraries I used**
        - Lucene
            - StandardAnalyzer
            - RAMDirectory
            - IndexWriter
            - IndexWriterConfig
            - Document
            - Field
            - TextField
            - DirectoryReader
            - IndexSearcher
            - QueryParser
        - PyLucene 8.8.1
        - Docker
        - A professional version of Pycharm IDE
        - Trec_eval
        - Xcode toolkit to run trec_eval-9.0.7

    - Python 3.9
        - Time
    - Packages:
        - wheel 0.36.2
        - setuptools 56.0.0
        - pip 21.1
        - JCC 3.9
        - lucene 8.8.1
- **Strengths and weaknesses of my design, and problems that myself and my system encountered**
    - The strength of this algorithm is, it’s fast in execution, *less than a minute in many runs*, because no major data structure is used or data stored
    - The weakness of the design is that it needs to be customized when using different datasets
    - The problems I have encountered are as follows:
        - I took me some time to understand the concept of the system and how it should work
        - It was not easy for me to learn Lucene, install it, and use it
        - Since Lucene is in Java and I haven’t worked with Java for more than 7 years, I had to use lucene wrapper for python, PyLucene where I am an active Python user
        - PyLucene wrapper has very few online support and generally bad ecosystem
        - The system encountered a very slow running time at the beginning because I was not saving documents sequentially; instead, I was trying to write them using one document “add” and that caused the program to crush
        - At the end, I was able to write documents on different fields of the document

2. **My customized new algorithm**
    - My new algorithm focused mainly on words similarity
    - Stop words deletion is a critical and essential part of the algorithm
    - To implement it, you need to do the following:
        - Read file of documents and queries
        - Index the documents using PyLucene: StandardAnalyzer() and RAMDirectory()

        - To start indexing, iterate through the lines of your documents, pick whatever information needed then send the content you want to use for search as a string to the stop words function so it deletes all unnecessary words that would slow down the running time and give better precision
        - After deleting unnecessary words, add the strings to a field in the RAM document
        - After adding all fields of documents, close the document writer and you will be done with the index step
        - You need to return the RAM directory and the Analyzer for the search step
        - In the search step, you will start iterating through the queries file to read and process them
        - As you did delete the stop words in the document file, you need to do the same and send your queries strings to the stop words functions to delete unnecessary words
        - After that you need to parse the queries by using QueryParser where you need to provide the field name of the document you wanna search and the analyzer
        - The next step will be scoring and ranking using the hits searched you decide (number of documents to return for each query searched)
        - The step after will be the last, writing the result into a file that has the trec format for performance evaluation
        - Using trec eval, you will get the metrics of your algorithm
    - No additional data has been used for this new algorithm

3. **Experimental results: the Precision, AvgPrec, and running time statistics; patterns I observed across the set of experiments**
    - Precision: Mean average precision (MAP): **0.1612**
    - AvgPrec: Average precision. Geometric mean (gm_map): **0.0543**
    - Please find trec_eval file in github below for full trec_eval metrics
    - Running time of only search: **1.7419092655181885** minutes
    - Running time of the algorithm using the HW data: **48.31597590446472** minutes
    - During working on the algorithm, I tried searching different parts of the documents, the more related sections I searched the better precision I got

- During building my stop words function to delete some keywords from the query, I realized that I get better precision when I delete the same words in both of the queries and documents files.
- Furthermore, I noticed that some stop words deletion make a really huge difference in improving the precision, while some other affected the precision negatively when deleting them

4. **What I have learned from this assignment**
    - I learned using Lucene and it’s Python wrapper, Pylucene
    - I learned using basics of docker to have an image of PyLucene, coady/pylucene and link the image to a professional version of PyCharm
    - I learned that I don’t need to store queries in a data structure but process and search them on the fly
    - I learned how to use indexing, parsing, some retrieval and ranking
    - I learned indexing document and writing them on the RAM using RAMDirectory() then reading the directory of documents to use it for search
    - I learned scoring retrieved documents

5. **You can find my CSE272HW1 repository on github:**
    - https://github.com/alexsalman/CSE272HW1/tree/master