Queen Mary University of London

COVID-19 Scientific Papers Search Engine Design

Module: ECS736P

Group 10:
Amir Sepehr Aminian, Bhavya Rajesh Makwana,
Alexander Sworski, Patrick Levermore

1. Problem statement and approach

The United Kingdom's (UK) National Institute for Health and Care Excellence (NICE) needs access to evidence on COVID-19 to develop rapid guidelines for healthcare professionals and prepare for future pandemics. The Medical Library Association curates a collection of COVID-19 papers in the CORD19 database. This paper proposes a search engine which allows users to (a) find research on specific areas which they may have missed during the fast flow of information during the pandemic and (b) understand what researchers focused on during different stages of the pandemic. This allows you to limit search results to COVID-19 articles where you can combine (using AND) these with other concepts to view articles about specific topics related to COVID-19 (e.g., AND COVID-19 terms with vaccine terms to find studies about vaccine research). This will help policy makers and medical professionals conveniently access research related to COVID-19. The scope of this project is to investigate and implement a variety of indexing and retrieval models for best search results, and for this task we decided to use CORD19 Dataset. We aim to search over a huge number of articles in quickest possible time and create a document ranking based on similarity functions like BM25.

2. Minimal Viable product (MVP)

We will develop a document retrieval model based on the CORD19 dataset using Python3, SpaCy and NLTK for the basic libraries and core NLP in the map reduce indexing. Indexing models will use map-reduce implementation, which will be utilized by the searcher that can score documents based on cosine similarity. Retrieval models will be based on TF-IDF, BM25 and BM25VA.

The MVP is to create a working annotation-based search engine with basic indexing and retrieval models and a simple user interface.

3. Future Enhancements

There can be charts and graphs search section added. The user searches a query and gets back relevant graphs possibly based on captions under the graphs in the documents.

There could be keywords in the search feature, such as ‘Author’. When this is followed by an author’s name, the search engine returns all papers by that author, filtered from the csv.

Papers could be inputted into the search engine, rather than short queries. The engine will then process all the keywords from this paper, with frequency, and use this to return similar papers to the user.

4. Dataset

CORD19, a collection of all published scientific papers on COVID-19 from https://ai2-semanticscholar-cord-19.s3-us-west-2.amazonaws.com/historical_releases.html. We use the 2020-05-12 version (~2GB). We initially use a subset of the dataset which will be annotated manually for the ground truth for search engine experimentation and evaluation of results. The dataset comes with captions that semantically captures the content of the document file along with the actual JSON converted format of PDF's. In addition to this, we use some benchmark techniques for the set of queries and relevance ranking can be obtained which can be used for relevance analysis.

5. Architecture

The research papers will be scraped from a JSON dataset, and the relevant information from the dataset will be first mapped to a SQL database, followed by an indexer, which creates an index for the data, which is also stored within the database. The user then uses a GUI to enter search terms, which then be used by a retrieval module to fetch the relevant documents from the database utilising the index created during import.

Following the initial description, the system consists of 7 main components, which are visualised in Figure 1. (Li et al., 2006; Singhal et al., 2010)

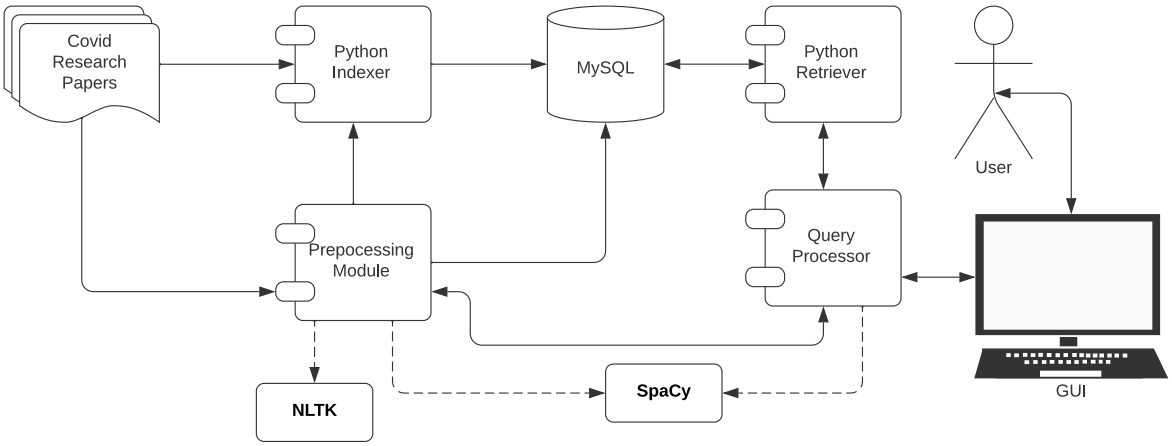

Figure 1: System architecture

The first component is the COVID-19 research JSON dataset. This is a fully passive component. In comparison to a classical search engine, this component is the equivalent of the web content crawler.

The first active component is the Python indexer, which calculates all the information required for indexing and adds it to the MySQL database. Simultaneously the pre-processing module processes the test using libraries such as NLTK before populating the database. The database architecture is described in the next chapter.

6. Query processor

The query processor will use a two-phase commit query processor model to access and join information from multiple data sources within the SQL. After taking the relevant terms from the query, it will use term-at-a-time process to apply each term to the retrieval model.

7. Retrieval model

The search engine will use the BM25 ranking function to assess document relevance to the search query. The BM25 system is a straightforward model with high performance, balancing term frequency and document length without adding too many parameters. BM25, often with some additions, still represents State-of-the-Art in document ranking and is used widely in search engines still.

BM25 uses a bag-of-words approach to track frequency of query terms within a document, without weighing for location. The search engine will use the formula shown below to calculate a score for each document based on the query, where Q = q_1, q_2, …, q_n represents keywords in the query, IDF (Inverse Document Frequency) is the inverse document frequency, and k1 and b are pre-optimized parameters.

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1\cdot\left(1-b+b\cdot\frac{|D|}{\text{avgdl}}\right)}$$

Equation 1: BM25

We can have two approaches towards retrieving relevant results for the user. We intend to focus on just the abstract to represent the document, for increased efficiency. However, if performance is notably better when using the whole article, we will implement this instead.

- Abstract based: In this approach, we remove the irrelevant columns, and we process only paper abstracts that are available in the CSV file. We put them all in the database and do the search based on the abstract and return the article URL or the article itself. Once the relevant abstracts are found, we then go through the corresponding articles that we scraped from the JSON to return more accurate results.
- Whole Article based: Here we scrape the JSON for the articles and index based on the whole document. Also removing the irrelevant columns when adding to the database.

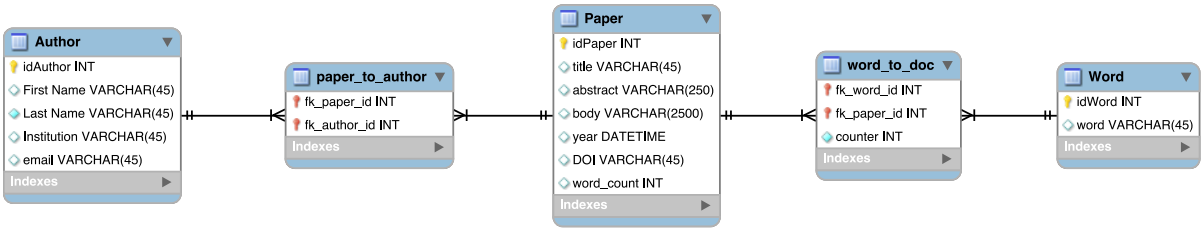

Figure 2: Database design

8. Database

The database (DB) architecture has been chosen to be normalised and relational, as the dataset we are working with is relatively small and dynamic changes are built within the architecture. Therefore, we are compromising speed in exchange for a relational and normalised database without redundancy. This allows for easy and quick changes to the content of the DB while it will be populated and allows for potential continuous updates, which would not be possible otherwise. In its current stage, as it can be seen in Figure 2, the database contains all variables necessary for TF-IDF in addition to pre-processed content, author, and publishing information. A further advantage of this fully normalised design is the possibility of easily adding fields and tables to accommodate potential alteration to the retrieval method.

We store following information in the database:

- Inverted index with the words as keys, and as save occurrence rate
- Word counter index which contains the length of all documents
- Word unique counter index which is needed for BM25VA

9. Relevance Analysis

We are going to use BEIR (Thakur et al., 2021), which already has evaluated systems on a Covid19 dataset. Therefore, we can use this framework to determine how well our system is responding to user's queries. Depending upon the nature and requirements of real-world applications, retrieval tasks can be either be precision or recall focused. To obtain comparable results across models and datasets in BEIR, we argue that it is important to leverage a single evaluation metric that can be computed comparably across all tasks. Decision support metrics such as Precision and Recall which are both rank unaware are not suitable. Binary rank-aware metrics such as MRR (Mean Reciprocal Rate) and MAP (Mean Average Precision) fail to evaluate tasks with graded relevance judgements. We find that Normalised Cumulative Discount Gain provides a good balance suitable for both tasks involving binary and graded relevance judgements.

10. Framework/Tools Employed

Tool/API	Description
GitHub	Version control and development collaboration
Python3/NLTK/SpaCy	Basic libraries will be used to implement search engine models and retrieval logics.
HTML, CSS, JavaScript	Front End Web Page Design to take input, display output & evaluation matrices
Codec	A codec is a device or computer program which encodes or decodes a data stream or signal. Codec is a portmanteau of coder/decoder
BeautifulSoup	Beautiful Soup is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

11. Product Development Backlog

The following are the product feature and technical backlog:

- Research document relevance/precision benchmark for CORD19 dataset
- Initial domain model design and architecture components research
- Configure document-search project, library dependencies and build scripts
- Document parser/processor to load data and index
- Map Reduce Indexing and database feasibility analysis
- Search/indexing methods and libraries analysis
- Stemming & Stop-word Removal
- Index model/framework to support flexible indexing implementations
- Query processor model/framework that will allow flexible query interpretation including query expansion
- TF-IDF, BM25, BM25VA retrieval model
- Relevance analysis components
- Experiment results and presentation

12. Roles/Responsibilities

All members of the team will be undertaking development and analysis activities. The team will adopt an agile delivery approach that fours developing product features incrementally. In addition, where possible the team will do pair programming.

The below are provisional role descriptions which are subject to change throughout product delivery:

Team Member	Roles
Alexander Sworski	System Structure/Architecture, Database setup, retrieval/indexing model development
Amir Sepehr Aminian	Repo Setup, Input Pre-processing, Designing user interface, Relevance Analysis
Bhavya Makwana	Tool Analysis/research, pairing on retrieval/indexing model development, preparation/analysis of results.
Patrick Levermore	Query processor model/framework, retrieval/indexing model development

References

Li, H., Councill, I., Lee, W.-C., Giles, C.L., 2006. CiteSeerx: an architecture and web service design for an academic document search engine, in: Proceedings of the 15th International Conference on World Wide Web - WWW '06. Presented at the the 15th international conference, ACM Press, Edinburgh, Scotland, p. 883. https://doi.org/10.1145/1135777.1135926

Singhal, Dr.N., Dixit, A., Sharma, A., 2010. Design of a Priority Based Frequency Regulated Incremental Crawler. International Journal of Computer Applications 1. https://doi.org/10.5120/23-131

Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., Gurevych, I., 2021. BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models. arXiv preprint arXiv:2104.08663