ASSIGNMENT III - SEARCH ENGINE IMPLEMENTATION - ECS736P - GROUP 51

Members:
Bineeta Kachhap (210619025)
Mokshith Kummari (210650383)
Krishna Sameer Kummetha (210470013)
Sanjay Ramesh (210811700)

PROBLEM STATEMENT AND APPROACH

Information retrieval conjointly works with Big Data. The basic concept of data retrieving in now equitable, and the focus has shifted to efficient retrieval. There are numerous model and approaches for data representation and retrieval respectively. The focus of this project is to understand the basic concept of indexing, searching and innovate on top of the basic model learned during curriculum

Clinical Decision Support Track/Okapi BM25

The Clinical Decision Support Track from 2014-2016 focuses on physicians looking for related biomedical articles to support their ongoing decisions. The project involves in understanding Full text retrieval. The acquired dataset comprises of medical cases, diagnosis and treatment received by the earlier patients. The Test Topics are created by medical experts as an instance of medical records. The purpose of this project involves developing a Text-based retrieval model that focuses of retrieving diagnosis, treatment, and testing decisions from the recorded dataset. Probabilistic approach with Okapi BM25 is the selected model that will be used during this assignment. Okapi BM25 is ranking function that shows the relevance of the document based on test query, this approach will be used to fetch the relevant clinical record based on test topics created by the medical experts

DATASET

Document Collection: For this project around 100+ GB of subset of PMC data has been taken for the search engine implementation.

2630847.nxml ×

article front article-meta article-id

```
10          <publisher>
11              <publisher-name>Molecular Diversity Preservation International</publisher-name>
12          </publisher>
13      </journal-meta>
14      <article-meta>
15          <article-id pub-id-type="pmid">19172194</article-id>
16          <article-id pub-id-type="pmc">2630847</article-id>
17          <article-id pub-id-type="doi">10.3390/md20080028</article-id>
18          <article-id pub-id-type="publisher-id">md-06-00550</article-id>
19          <article-categories>
20              <subj-group subj-group-type="heading">
21                  <subject>Article</subject>
22              </subj-group>
23          </article-categories>
24          <title-group>
25              <article-title>A Tropical Marine Microbial Natural Products Geobibliography as an
26                  Example of Desktop Exploration of Current Research Using Web Visualisation
```

DATASET

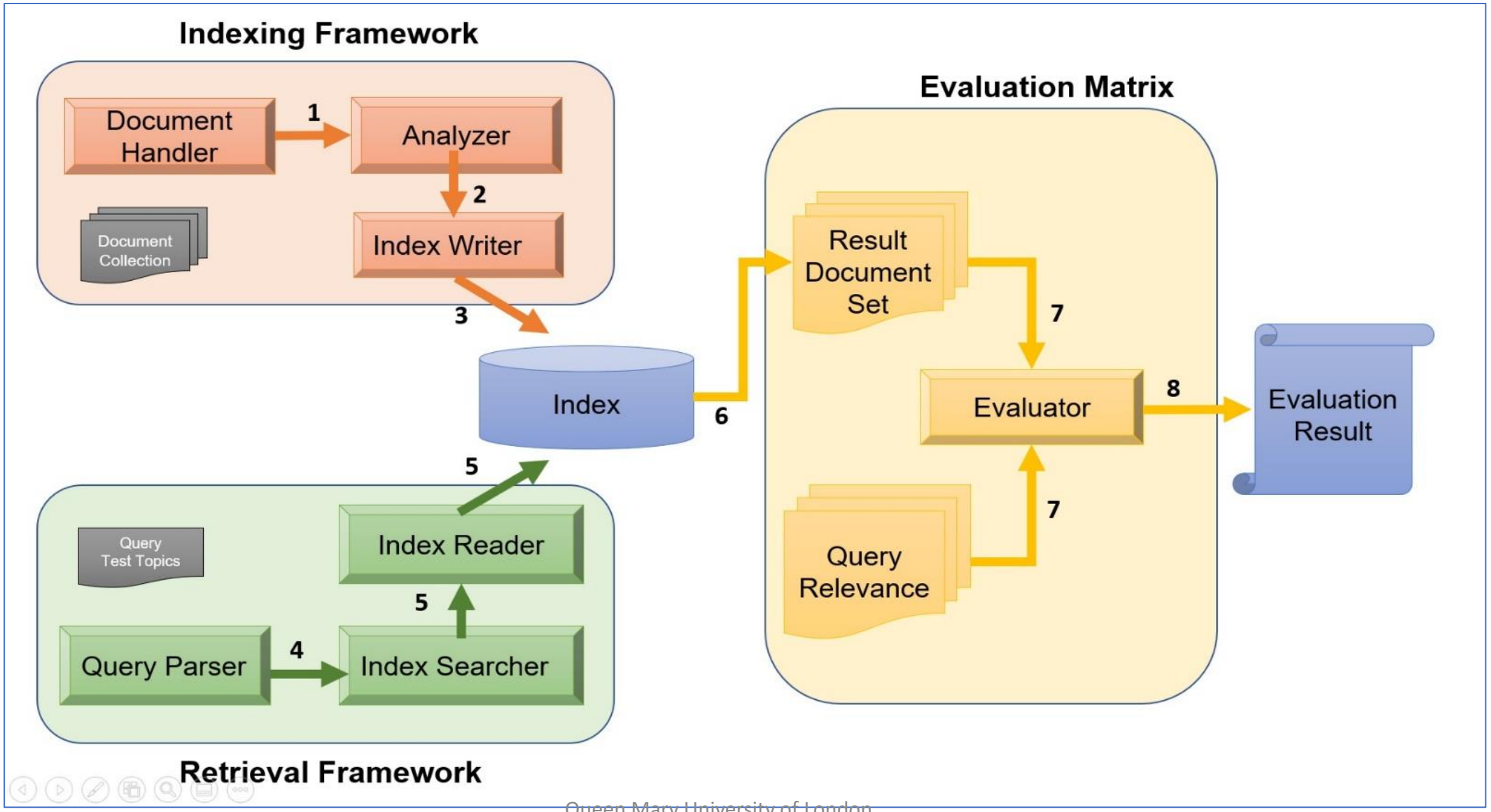
Test topics: The summary and description nodes in the test topic will be used as the search query.

```
−<topic number="2" type="diagnosis">
  −<description>
      A 62 yo male presents with four days of non-productive cough and one day of fever. He is on immunosuppressive medicat
      bronchoalveolar lavage (BAL). BAL fluid examination reveals owl's eye inclusion bodies in the nuclei of infection cells.
   </description>
  −<summary>
      A 62-year-old immunosuppressed male with fever, cough and intranuclear inclusion bodies in bronchoalveolar lavage
   </summary>
```

Query Relevance : The relevance of 1 and 2 has been considered as relevant documents for validation. The decision was taken to validated if the implemented model can correctly search the document.

```
11 0 3643613 0
11 0 3643833 2
11 0 3644651 0
11 0 3644769 2
11 0 3644779 0
11 0 3647563 1
11 0 3648368 0
11 0 3648442 0
11 0 3649594 0
11 0 3650978 0
11 0 3651721 0
```

ARCHITECTURE



ARCHITECTURE

Indexing Framework :

- A python script was written to process 100GB of clinical records → reduced to ~4GB.
- Output as .parquet format → less space and feasible to read both in Python or Java.
- Parallel processing to read all the files.

- *<article-id "pub-id-type" = pmc>*

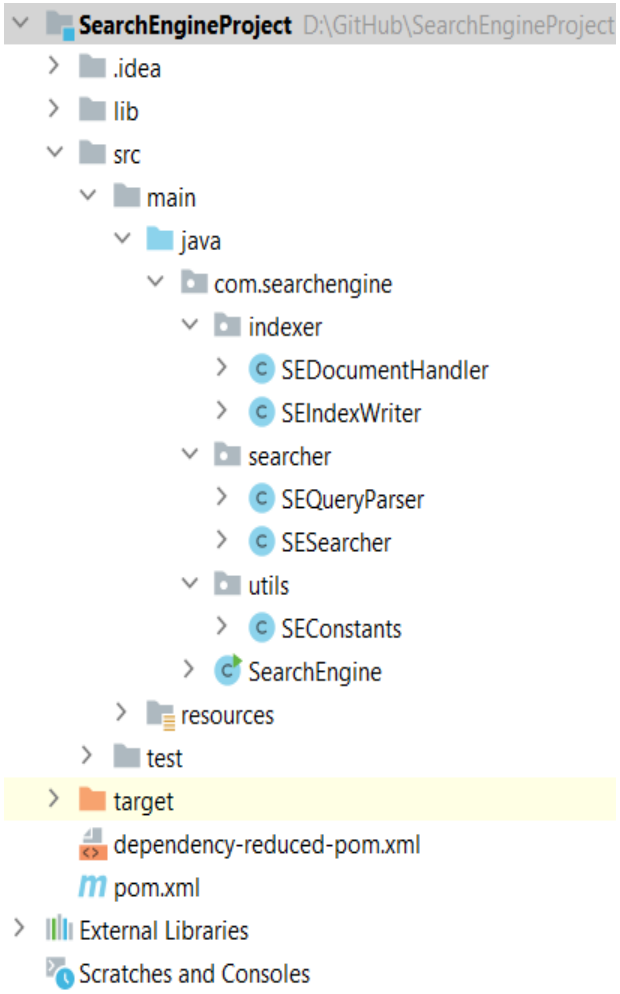
unique document id used further implementation for identification

- *<journal-title>* : Short description
- *<p>* : the content of clinical trails and decision done for existing cases and recorded.

```
p = Pool(50)
daas = p.map(getCorpusData, files)
corpusData += daas
p.terminate()
p.close()
```

```
try:
    tree = ET.parse(file)
    root = tree.getroot()
    for tag in root.iter('article-id'):
        if(tag.attrib.get('pub-id-type') == 'pmc'):
            pmcId = ((file.split('\\')[7]).replace('.nxml', '')) if tag.text

    '''for tag in root.iter('journal-id'):
    journalId = tag.text'''

    for tag in root.iter('journal-title'):
        journalTitle =tag.text

    data = ''
    for tag in root.iter('p'):
        data += '' if tag.text is None else tag.text

    return (pmcId, journalTitle, data)
except:
    return ''
```

ARCHITECTURE



- SEDocumentsHandler: Had planned to include the preprocessing in here but it was moved to python Script.
- SEIndexer: Custom indexer to read all the .parquet files and create document index. *org.apache.lucene.index.IndexWriter* was used.
- SEQueryParser: Custom query Parser that takes the topicQuery number as input and return the parsed query of either <description> or <summary>. javax.xml.parsers.DocumentBuilder to read the topics-2015-A.xml file.
- SESearcher: custom searcher that takes the parsed query from the SEQueryParser and retrieves information from the indexed files with BM25Similarity.
- SearchEngine: main() java entry point for user to either create index or enter query number along with number of top documents.

Demo

- Dataset
- Pre-processing
- Indexing
- Retrieval
- Evaluation

EVALUATION

- **Precision** : Basically, this metric quantifies how many items in the top-K results were relevant. Mathematically, precision the number of true positives divided by the number of true positives plus the number of false positives.

$$Precision@k = \frac{true\ positives@k}{(true\ positives@k) + (false\ positives@k)}$$

- In our case the True positive is the number of relevant documents retrieved and false positives are the number of non-relevant documents retrieved.

EVALUATION

- Recall : This metric gives us how many actual relevant results were retrieved out of all actual relevant results for the query. Mathematically it is expressed as

$$Recall@k = \frac{true\ positives@k}{(true\ positives@k) + (false\ negatives@k)}$$

- The above formulae generally tells us the number of relevant documents retrieved out of the total number of relevant documents.

ISSUES FACED

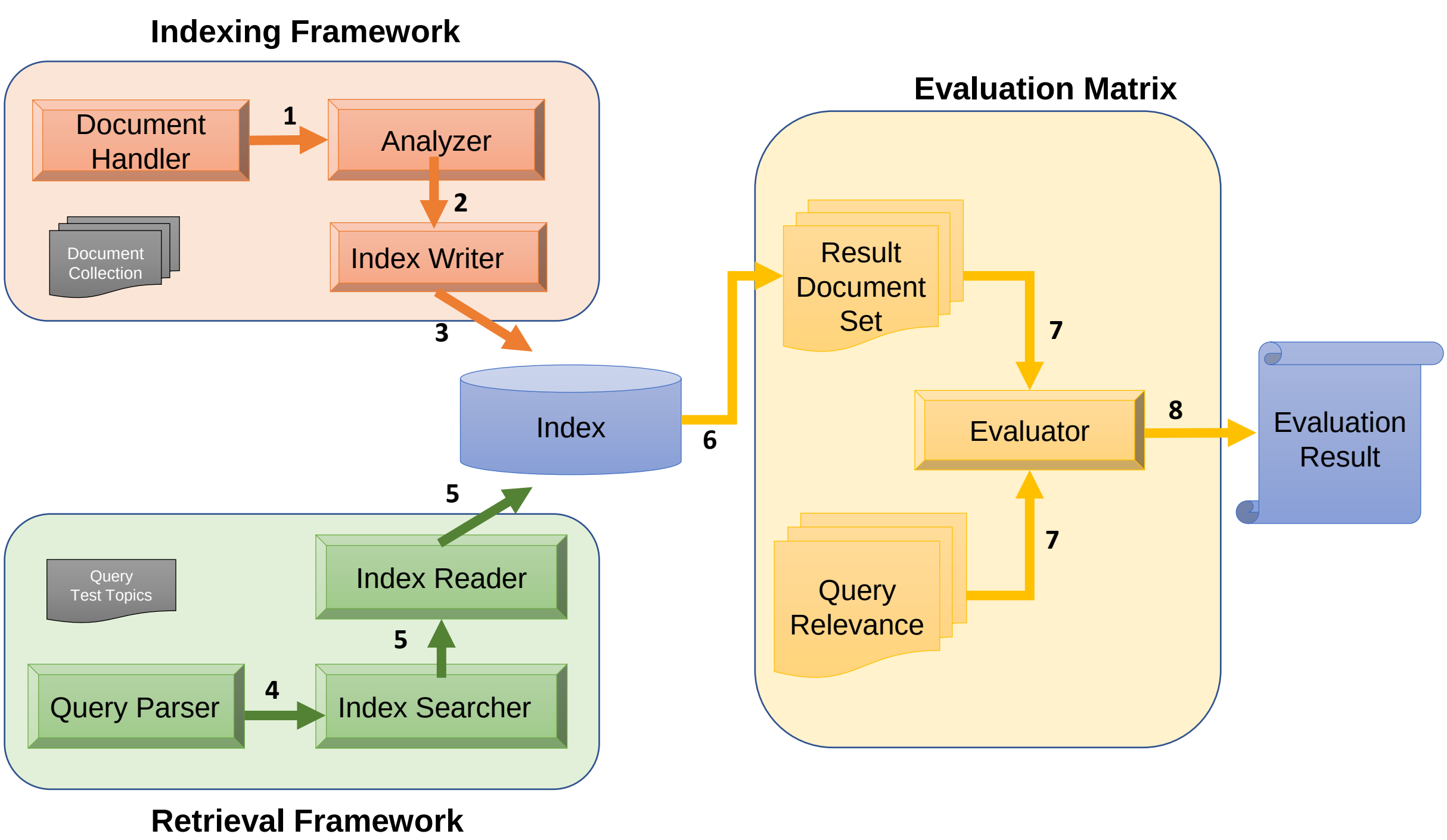
- **Storage**: The initial dataset .tar collected was of 10GB. expectation was that the unzipped data might be around 40-50 GB but it was 100GB.
- **Processing Unit**: To process 100 GB of data the colab was unreliable as we had to parallel process the data.
- **pyLucene**: The initial plan was to build the project in python, but PyLucene uses wrapper class to access Java Lucene-Core, hence the project was moved to Java based implementation.
- **Indexing**: One of the document file had details missing, hence had to re-work on Indexing framework to understand the data. Multiple Analyzer were tested to get the relevant document which led to repeated execution of the framework.
- **Lucene Library**: Few APIs and classes are now deprecated in Lucene 9.1.0, throughout the development process learnt the latest library feature.
- **Searching / Retrieval**: Though the retrieval process result in document with the ranking score. the framework doesn’t not give correct list of relevant document. Further analysis needs to be done to understand.

FURTHER IMPROVEMENT

- **Indexing**: Merge document processing and Indexing rather than having two separate steps.
- **Custom Analyzer:** Custom tokenizer and multiple token filter could be implemented.
- **Evaluation/Validation**: Investigation on current model.
  - Appropriate Similarity measure to use.
  - Analyzer(Tokenizer and Token Filter) used to index and search for text retrieval pertaining to clinical linguistic data.

Thank You.

Members:
Bineeta Kachhap (210619025)
Mokshith Kummari (210650383)
Krishna Sameer Kummetha (210470013)
Sanjay Ramesh (210811700)


Indexing Framework
Document Handler
1
Analyzer
2
Index Writer
Document Collection
3
Index
6
Evaluation Matrix
Result Document Set
7
Evaluator
8
Evaluation Result
Query Relevance
7
5
Index Reader
5
Query Test Topics
Query Parser
4
Index Searcher
Retrieval Framework