# Information Retrieval Programming Task P01

**Team**
Alishiba D'souza
Neetha Jambigi
Rutuja Shivraj Pawar
Tirtha Chanda

10.12.2017

**Abstract**

The Programing task P01 is defined to develop our own Information Retrieval system using Apache Lucene version 7.1.0 and Java. This document gives an overview of the concerned Information Retrieval system developed and explains as to how it satisfies different criteria's mentioned for the developed system.

## 1 Description of the Program

According to the different criteria's specified for the developed system, different methods are modeled in the Program Structure.

- **Using Lucene, parse and index HTML documents that a given folder and its subfolders contain. List all parsed files.**

  Lucene is an open source search library that provides the standard functionality for analyzing, indexing, and searching of text-based documents.

- **Consider the English language and use a stemmer for it (e.g. Porter Stemmer).**

  Porter Stemmer is used for term normalization process carried out in Information Retrieval Systems. The Porter Stemmer is used to remove the commoner morphological and inflexional endings from English words.

- **Select an available search index or create a new one (if not available in the chosen directory).**
- **Make possible for the user to choose the ranking model, Vector Space Model (VS) or Okapi BM25 (OK).**

  Ranking is a process of sorting the documents from a collection so that the best relevant documents in the form of best results are displayed to the user at the earliest in the result list.

  The Ranking Model, Vector Space Model an algebraic model, represents a natural language document in a formal way with the use of vectors in a multi-dimensional space. It also allows decisions to be made as to which documents are similar to each other and also to the queries fired by the user.

  Ranking Model, Okapi BM25 (OK) is based on a probabilistic retrieval framework, that ranks a set of documents based on the specified user query terms appearing in each document. In the process of ranking, it does not take into consideration the inter-relationship between the user query terms within the document.

- **Print a ranked list of relevant articles given a search query. The output should contain 10 most relevant documents with their rank, title and summary, relevance score and path.**

The Ranking List is prepared on the basis of the output from the Ranking model for a given user search query and it displays the most relevant documents with their rank, title, relevance score and path

- **Search multiple fields concurrently (multifield search): not only search the document's text (body tag), but also its title.**

## 2 Structure of the Program

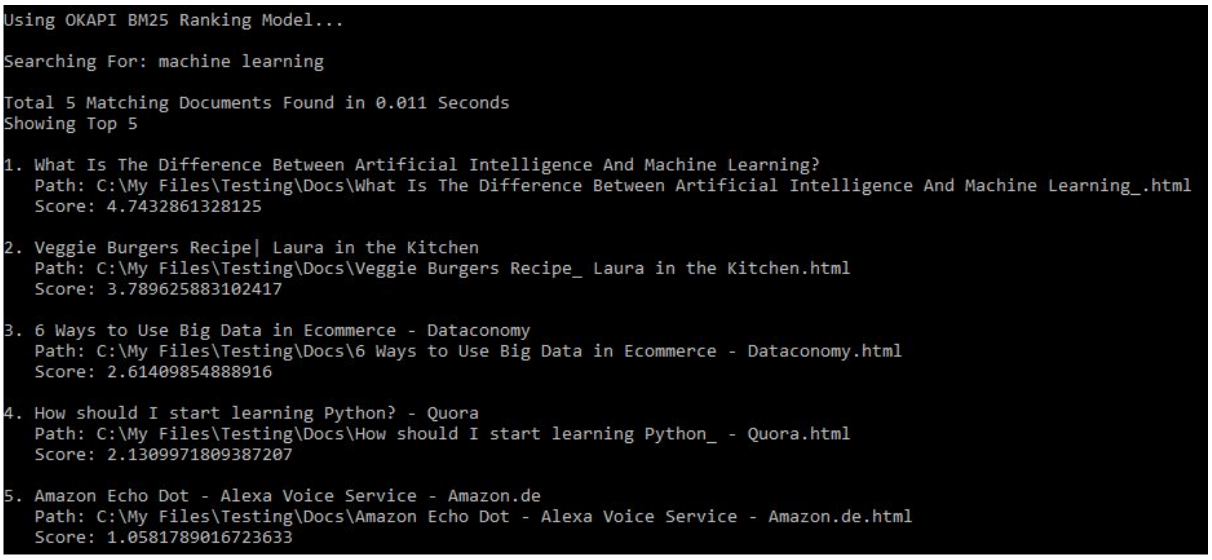
- **Main.java :** The Main Program takes the command line arguments from the user. After checking for its validity, it retrieves the Document and the Index Path from the command line argument. The class IndexFiles.java is then invoked to run the Indexer. The class SearchFiles.java is also invoked to run the Searcher for the input.
- **IndexFiles.java :** This Indexer class takes document and the Index folder path as as input. It then checks if the specified document directory exists and whether the JVM has permissions to read files in this directory. If this step is successful, it then takes the system time when the indexing starts so that the number of seconds required to index the documents can be calculated.
  - **Analyzer analyzer = new StandardAnalyzer() :** A StandardAnalyzer object is initialized. This will convert tokens to lowercase and also filters out stopwords.
  - **IndexWriterConfig iwc = new IndexWriterConfig(analyzer) :** This is used to store all the configuration parameters for IndexWriter.
  - **IndexWriter writer = new IndexWriter(dir, iwc) :** This is used to create and maintain an index.
  - **indexDocs(writer, docDir) :** This is used to start the Indexing process. This method crawls the document path folder and then creates document objects with all the files inside it. The document object instances are then added to the IndexWriter. This method calls indexDoc which is used to Index a document.
  - **indexDoc(IndexWriter writer, Path file, long lastModified) :** This method is used to Index a document. The parseHTML and doStemming methods are invoked in indexDoc.
  - **parseHTML(Path file) :** This method creates a Scanner object to read all the lines from the html file. The raw html contents get stored in the String rawContents. Jsoup then parses the rawContents and creates a parsed document and then returns it.
  - **doStemming(String parsedContents) :** This method is used to perform stemming on the extracted html contents.
- **SearchFiles.java :** This Searcher class takes Index Path, Query and Ranking Model as an input.
  - **IndexReader reader = DirectoryReader.open(FSDirectory.open(Paths.get(indexPath))) :** This initializes the Index Reader.
  - **IndexSearcher searcher = new IndexSearcher(reader) :** This initializes the Index Searcher.
  - The validity of the Ranking model is verified accordingly. If the validation is successful, then the similarity is set accordingly for the Ranking model provided.
  - **MultiFieldQueryParser mfparser = new MultiFieldQueryParser(fields, analyzer) :** This is used to search multiple fields in the document objects using the same parser instance.
  - **Query query = mfparser.parse(q) :** This parses and stores the Query in a Query object.
  - **doPagingSearch(BufferedReader in, IndexSearcher searcher, Query query, int maxHitsDisplay) :** This method is used to execute the actual search process.

## 3 Program Execution

The jar file of the program can be executed from the command line as follows:

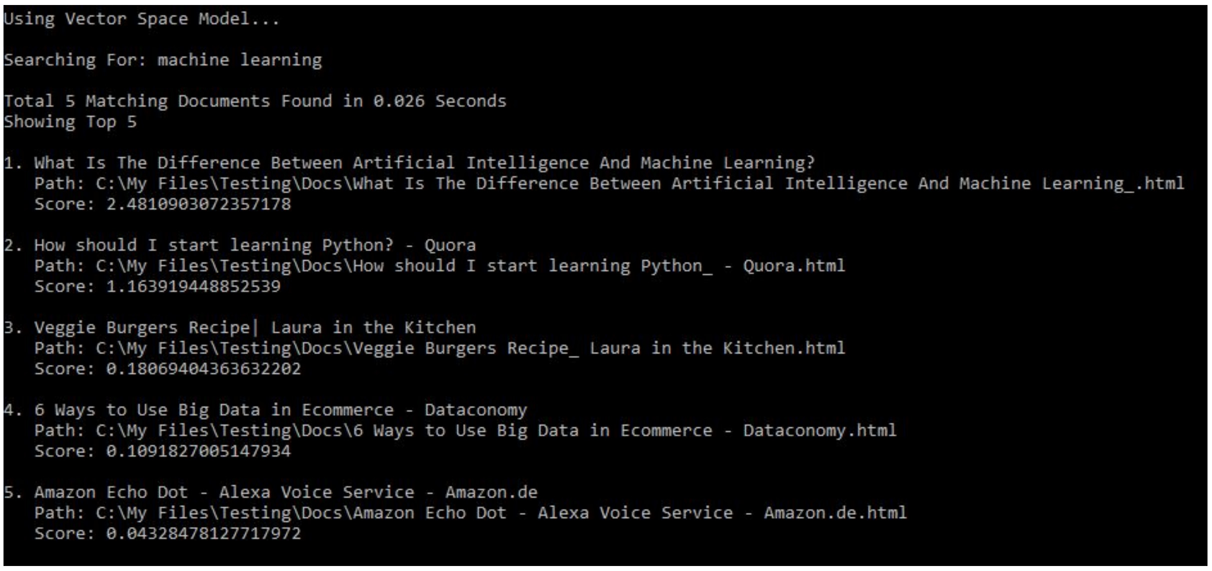
```
java -jar [jar file name] [path to document folder] [path to index folder] [VS/OK] [query]
```

Below is the comparison of the obtained output:



```
Using OKAPI BM25 Ranking Model...

Searching For: machine learning

Total 5 Matching Documents Found in 0.011 Seconds
Showing Top 5

1. What Is The Difference Between Artificial Intelligence And Machine Learning?
   Path: C:\My Files\Testing\Docs\What Is The Difference Between Artificial Intelligence And Machine Learning_.html
   Score: 4.7432861328125

2. Veggie Burgers Recipe| Laura in the Kitchen
   Path: C:\My Files\Testing\Docs\Veggie Burgers Recipe_ Laura in the Kitchen.html
   Score: 3.789625883102417

3. 6 Ways to Use Big Data in Ecommerce - Dataconomy
   Path: C:\My Files\Testing\Docs\6 Ways to Use Big Data in Ecommerce - Dataconomy.html
   Score: 2.61409854888916

4. How should I start learning Python? - Quora
   Path: C:\My Files\Testing\Docs\How should I start learning Python_ - Quora.html
   Score: 2.1309971809387207

5. Amazon Echo Dot - Alexa Voice Service - Amazon.de
   Path: C:\My Files\Testing\Docs\Amazon Echo Dot - Alexa Voice Service - Amazon.de.html
   Score: 1.0581789016723633
```

Figure 1: Output Result for the search string machine learning using OK.



```
Using Vector Space Model...

Searching For: machine learning

Total 5 Matching Documents Found in 0.026 Seconds
Showing Top 5

1. What Is The Difference Between Artificial Intelligence And Machine Learning?
   Path: C:\My Files\Testing\Docs\What Is The Difference Between Artificial Intelligence And Machine Learning_.html
   Score: 2.4810903072357178

2. How should I start learning Python? - Quora
   Path: C:\My Files\Testing\Docs\How should I start learning Python_ - Quora.html
   Score: 1.163919448852539

3. Veggie Burgers Recipe| Laura in the Kitchen
   Path: C:\My Files\Testing\Docs\Veggie Burgers Recipe_ Laura in the Kitchen.html
   Score: 0.18069404363632202

4. 6 Ways to Use Big Data in Ecommerce - Dataconomy
   Path: C:\My Files\Testing\Docs\6 Ways to Use Big Data in Ecommerce - Dataconomy.html
   Score: 0.1091827005147934

5. Amazon Echo Dot - Alexa Voice Service - Amazon.de
   Path: C:\My Files\Testing\Docs\Amazon Echo Dot - Alexa Voice Service - Amazon.de.html
   Score: 0.04328478127717972
```

Figure 2: Output Result for the search string machine learning using VS.

Given a search query "machine learning," both VS and OKAPI models retrieved 1 completely irrelevant document and 1 slightly irrelevant document. However, the VS model ranked the irrelevant document at rank 3, while the OKAPI model ranked the irrelevant model at rank 2. - screen shots above indicate this.

## 4 Observations:

Both VS and OKAPI models performed similarly for 10 instances of a search query ”machine learning”. The times taken are graphically depicted in the bar chart above.

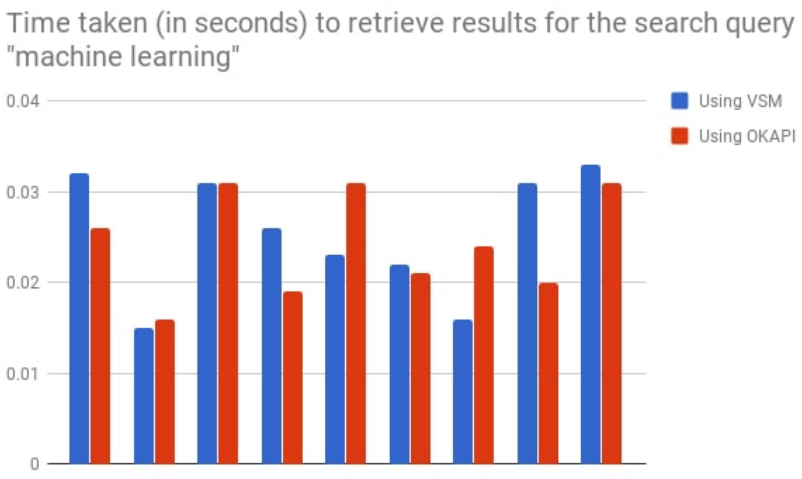


Figure 3: comparison of execution time vsm and Okapi.