Basic and Advanced Ranked Retrieval Techniques using Java

Ryan Chew, Jin Zeng

Royal Melbourne Institute of Technology

Author Note

The implementations described in here were done using Java 12 without the use of external libraries. All instructions for running the files are located in the README.

Table of Contents

[Abstract 3](#_Toc21238802)

[Basic and Advanced Ranked Retrieval Techniques using Java 4](#_Toc21238803)

[Okapi BM25 Ranked Retrieval 4](#_Toc21238804)

[Implementation 4](#_Toc21238805)

[Indexing. 5](#_Toc21238806)

[Querying Process. 5](#_Toc21238807)

[Data Structures and Algorithms. 6](#_Toc21238808)

[Min-Heap 6](#_Toc21238809)

[Accumulator 6](#_Toc21238810)

[Advanced IR Feature 7](#_Toc21238811)

[Automatic Query Expansion 7](#_Toc21238812)

[Achievement Goals 7](#_Toc21238813)

[[Heading 4] 7](#_Toc21238814)

[*References* 8](#_Toc21238815)

[Footnotes 9](#_Toc21238816)

[Tables 10](#_Toc21238817)

[Figures 11](#_Toc21238818)

Abstract

[The abstract should be one paragraph of between 150 and 250 words. It is not indented. Section titles, such as the word Abstract above, are not considered headings so they don’t use bold heading format. Instead, use the Section Title style. This style automatically starts your section on a new page, so you don’t have to add page breaks. To apply any text style in this document with just a tap, have a look at Styles on the Home tab of the ribbon.]

Keywords: Search Engine, Automatic Query Expansion, Okapi BM25

Basic and Advanced Ranked Retrieval Techniques using Java

# Okapi BM25 Ranked Retrieval

Okapi BM25 (BM25) is a probabilistic ranked retrieval function that was developed for use in estimating relevance of documents in a collection with regards to a given search query. The version of BM25 that was implemented in this program was simplified to:

The terms are defined as:

* Q is a query consisting of terms t
* Dd is a document in the collection
* N is the number of documents in the collection
* fd,t is the number of occurrences of t in d
* ft is the number of documents containing term t
* Ld and AL are the document length and average document length, measured as the number of characters in the data file.
* k1 and b are parameters that are initially be set as k1 = 1.2 and b = 0.75

## Implementation

The implementation of BM25 in this program was done using the standard Java *Math Package* (Oracle, n.d.). Reading the supplied equation, an implemented equation was created with the required variables supplied via the arguments. Changes were required to be made to the Indexing files to record the size of each individual document in the mapping file. This was done to correctly implement the BM25 function for each document.

Indexing. The changes to the indexing modules were required to perform the calculation of BM25. Changes included the recording of document content lengths, including the data between the ‘heading’ and ‘text’ tags in the original document collection. These values were recorded together with the mapping information of the raw document identifiers and the uniquely generated identifiers of the indexing program (Chew, 2019).

### **Querying Process.**

Stopping. The process that query terms were subjected to firstly include a stopping of the terms, ensuring the format of the query terms are consistent with the indexed documents. This process employs a method originally created for indexing but has been utilized here in the same fashion for the query terms.

File Handling. Next the lexical and mapping data are read from their respective files into memory of the program. This is done using standard Java IO, similarly done to the indexing portion of the program. The main different in this implementation would be the extra recording of the document weights, as lengths of the document and the averaging of the weights.

Similarity Scoring. After the intake of the required inverted list information, the query processing module is created using the inverted list data. Post-creation, the query terms are then processed one-by-one through the documents and calculations to find all relevant documents for the term and to calculate the partial accumulator scores for the documents. The process in which this has been done is through simply iterating through each term and running the process with the term. Over each term the similarity score is then accrued, run through a min-heap, and finally returned to the main function to be presented to the user.

## **Data Structures and Algorithms.**

Min-Heap. Data structures of interest in this implantation would involve the use of the Priority Queue as an implementation of a min-heap. This was used as a simple method of sorting the data organically without the use of a dedicated sorting function.

Accumulator. Another data structure that is crucial to the functionality of the program is the implementation of the Accumulator (util.Accumulator). In practice this structure is purely a container of data, however to allow the structure to be compatible with the Priority Queue, there needed to be a proper definition of being able to determine which accumulator was of higher value. In this case, this was achieved by extending the Comparable class and overriding the ‘compareTo’ method. This implementation of the accumulator allowed for the internal heapification of the inputted accumulators in the Priority Queue.

Advanced IR Feature

# Automatic Query Expansion

Automatic Query Expansion (AQE) defines a method in which the original query is reformed to a state that includes a higher number of query terms to search. This is a form of pseudo-relevance feedback where it is assumed that documents that are relevant to the query will contain other terms related to the original search query.

## Achievement Goals

The aim of this feature is to increase the effectiveness of the Information Retrieval system whereby the returned documents are ranked more effectively so that documents that were initially passed as not relevant that in reality have higher relevancy are returned.

## Implementation

This feature was implemented by building upon the initial BM25 program. The separation of methods was achieved using command line arguments.

Indexing. For this implementation, additional changes were needed to be done to the indexing module of the program. Specifically, locations of document positions in the original collection were needed to be recorded (DocumentHandler: 115, DocumentHandler: 143). This was to allow for AQE to be done with pseudo-relevance feedback.

Querying Process. The changes to the search function included a second pass of query terms after expansion. The expansion was performed using a TSV calculation:

Using this calculation in a min heap structure, the most relevant terms were selected. After selecting the terms, the new query terms are passed through the ranked searching algorithm again. This time, the new query terms’ similarity score were calculated using the Robert-Sparck Jones score.

Using this measure, the similarity scores are calculated. After which the top accumulators will be selected. And returned to the user with the original queries.

[Heading 5]. [Like all sections of your paper, references start on their own page, as shown on the page that follows. The body of the References section uses the Bibliography style. For more detailed information on formatting references, see the APA Style Manual, 6th Edition. (Surname, Year)

# *References*

Claudio Carpineto, G. R. (2012, January 1). A Survey of Automatic Query Expansion in Information Retrieval. *ACM Computing Surveys (CSUR), 44*(1), 1-50.

Olalere A. Abass, O. F. (2017, July 5). Automatic Query Expansion for Information Retrieval: A Survey and Problem Definition . *American Journal of Computer Science and Information Engineering , 4*(3), 24-30.

Oracle. (n.d.). *java.math (Java Platform SE 7)*. Retrieved from Oracle Javadocs: https://docs.oracle.com/javase/7/docs/api/java/math/package-summary.html

Footnotes

1[Add footnotes, if any, on their own page following the references. For APA formatting requirements, it’s easy to just type your own footnote references and notes. To format a footnote reference, select the number, then apply the Footnote Reference. The body of a footnote, such as this example, uses the Normal text style. (Note: If you delete this sample footnote, don’t forget to delete its in-text reference as well.)]

Tables

Table 1

[Table Title]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Head | Column Head | Column Head | Column Head | Column Head |
| Row Head | 123 | 123 | 123 | 123 |
| Row Head | 456 | 456 | 456 | 456 |
| Row Head | 789 | 789 | 789 | 789 |
| Row Head | 123 | 123 | 123 | 123 |
| Row Head | 456 | 456 | 456 | 456 |
| Row Head | 789 | 789 | 789 | 789 |

Note: [Place all tables for your paper in a tables section, following the references (and, if applicable, the footnotes). Start a new page for each table, include a table number and table title for each, as shown on this page. All explanatory text appears in a table note that follows the table, such as this one. Use the Table/Figure style to get the spacing between table and note. Tables in APA format can use single or 1.5-line spacing. Include a heading for every row and column, even if the content seems obvious. To insert a table, on the Insert tab, tap Table. New tables that you create in this document use APA format by default.]

Figures



Figure 1. [Include all figures in their own section, following references (and footnotes and tables, if applicable). Include a numbered caption for each figure. Use the Table/Figure style for easy spacing between figure and caption.]

For more information about all elements of APA formatting, please consult the APA Style Manual, 6th Edition.