8/4/2019

Ryan Chew (s3714984), Jin Zeng (s3688213)

RMIT University

An examination of inverted lists and search querying

An implementation

# **Introduction**

The action of searching through thousands of documents for a single word in a term is a simple concept, one so simple, that it should not be thought of to be very complicated. Furthermore, this should all be done no longer than a fraction of a second. A huge amount of research and development has gone into creating faster ways to complete this task. Here, the focus is on developing a basic inverted index and querying program.

An *inverted index[[1]](#footnote-1)* is a data structure of documents, consisting of words, which documents and the frequency of the words in the documents. Though many implementations of inverted indexes may also contain information on the location of the words in each document, it was not implemented in this version.

This paper discusses the decisions made in the implementation of the inverted index and related modules, and the querying module.

The rest of the paper is organized as follows. The next section discusses index construction and the decisions made in regard to handling and processing the text data. In Section 3, the stoplist module will be described, as well as implementation thoughts. Section 4 will focus on the query mechanism and how the function works, and the direction taken. Section 5 explores the size difference between the produced files and the original documents.

# **Index Construction**

This section will contain explain methods and decisions pertaining to document parsing methods. How the data is merged together for further processing to the construction of the inverted index. The formatting of the data written to disk.

## Tokenization of terms

The tokenization and normalization of terms is quite important in the fast and efficient indexing, querying of data.

The method used to process the data is very simple. During the progression of reading the lines of the given collection of documents, the normalization of the text was performed in parallel. Using the available methods in the java standard library, a set Regular Expressions (regex) term was constructed to efficiently parse out punctuation and standard word contractions (aren’t -> are not). This set of regex terms were applied to the inputted header and body text for processing prior to storage.

After the text has been processed, the program will create an internal document for further processing.

## Statistical frequency gathering

A key step in the creation of inverted index lists is the gathering of the frequency of terms in the document collection.

The implementation of the gathering mechanism in the program employs a HashMap, used for its more efficient non-synchronized nature. The program iterates through a list of given text data and checks if the hashmap contains a term. If that term does not exist it will create a new entry for the term in the hashmap, or if that entry does exist but not the document in that terms collection of documents, it will create a new entry for the document under that term. Otherwise, it will increment the current counter for that document under that term. This implementation of the hashmap allows for quick storage and retrieval of the words and the document data. After the collection of terms from the documents, the empty string term is removed.

# Bibliography

Black, P. E., 2017. *Inverted Index.* [Online]   
Available at: https://www.nist.gov/dads/HTML/invertedIndex.html  
[Accessed 4 August 2019].

1. (Black, 2017) [↑](#footnote-ref-1)