

# Inverted index<sup>1</sup> with dictionary file and position index

Alexander Ries

Philipp Steiner

Information Retrieval – Aufgabe1  
(Übung), WCIS 2014 FH Kufstein Tirol

## ABSTRACT

This paper describes the application of the inverted index with the use of position index for documents and its textual content.

## Categories and Subject Descriptors

H.3.1 [Content Analysis and Indexing]: Indexing methods

## General Terms

Algorithms, Performance

## Keywords

Information retrieval, text processing, inverted index.

## 1. INTRODUCTION

Information retrieval is an increasingly important topic in today's world as it becomes more difficult every day to find relevant information. The increase of available information online requires the development and improvement of new and existing concepts to efficiently find and retrieve relevant information. This paper will implement an inverted index with two different algorithms in order to compare the use of different technics in terms of performance and efficiency when processing large amount of text.

## 2. OBJECTIVE

The purpose of this project is to demonstrate the ability of certain algorithms to process a number of text files and create an inverted index for those files and words. In order to gain more insight into the importance of the algorithm used to index text files the application will use two different variants of indexing text files and compare their different technics.

## 3. PROGRAM & ALGORITHM

The application was developed with a minimum required user interface in order to execute the required tasks. First the user can choose between Exercise 1 and 2.

In the first exercise the user is furthermore prompted to chose between 2 variants of creating an inverted index of the documents in the Reuters folder. For the purpose of efficiency the inverted index is only available in memory and not in a persistent context (expect dictionary file). After the successful creation of the index the user is able to search a term in the index and get a list of locations for the term if it is contained in any of the indexed files.

In the second exercise, the files from the ClueWeb09 Collection get first downloaded. Afterwards the program reads though all the files and creates an inverted index out of them, using variant 2 of exercise 1. Afterwards the user can search a term in the files. The program shows all files and positions within the files the term is found.

### 3.1 Variant 1

This algorithm is based on the standard process of creating an inverted index including the position index for each term. The

algorithm will first extract all the terms from each file and then sort the terms in alphabetical order. The next step will iterate through the files again and extract the position for each occurrence of each term and save the result in a HashMap.

### 3.2 Variant 2

This algorithm is substantially different from variant 1. This variant will scan each file and extract the terms for each file. The terms are then individually compared to any existing occurrence of the term in the index and either update the term with a new position or create the index with its initial position respectively. The result for each file is an updated HashMap of terms and their positions that will then be used as input for the next file.

### 3.3 Dictionary file

The dictionary file is a text file that is created storing all the terms retrieved from the documents in a separated line. Because variant 2 never has a complete list of words in an ordered format only variant 1 is able to create such a file. The directory that the file will be created is shown in the value of the variable "dictionary\_directory" in the application.

### 3.4 Test Data

The data set used for testing in exercise 1 is from Reuters and is available under the following link:  
<http://www.daviddlewis.com/resources/testcollections/reuters21578/>

For exercise 2 the test data from the ClueWeb09 Collection is used which consists of links to collected websites. The was stored in a WARC files which was read directly in java code in the program.

### 3.5 Handling WARC Files

To use the data in the warc files the program uses two java classes referenced at the ClueWeb09 Collection website. WarcHTMLResponseRecord and WarcRecord. With these sources it is possible to read though all the entries in a warc file directly in java. The programm extracts the urls and downloads them with the JSOUP Framework. JSOUP is a Web Crawling Framework and the jar files can be found at this under <http://jsoup.org/packages/jsoup-1.8.2.jar>.

## 4. Conclusion

### 4.1 Results

The variants 1 & 2 were both tested with the same set of data consisting of 21,578 articles roughly 28 MB in size. Variant 1 of the application was not able to create the inverted index of the whole data set in a reasonable provided time. The indexing of smaller files shows that this variant is able to index approximately 573 terms/second. Variant 2 was able to create the inverted index of the given data set with position index in less than 3 seconds.

## 4.2 Performance

The two different variants show quite a substantial difference in performance and efficiency. Variant 1 seems very slow compared to variant 2 and part of the reason is that the files have to be scanned several times for each individual process stage (extraction, sorting, position index). Variant 2 only scans each file once as it combines all the stages of text processing in one sequence. Further investigation into the algorithms could lead to improved

efficiency of either variant which might lead to better performance and efficiency.

## 5. REFERENCES

- [1] Manning, Christopher D, Prabhakar Raghavan, and Hinrich Schütze. Introduction To Information Retrieval. New York: Cambridge University Press, 2008. Print.