University *of Ljubljana*
Faculty *of Computer and Information Science*

Programming Assignment 3: Data Indexing and Retrieval

Tim Vučina

Abstract

We present our implementation for the course project and roughly outline the problems and solutions during the development. We analyze and show a general approach to parsing and extracting textual context from the HTML files as well as storing the gathered data into an index which is then used for querying and data retrieval.

Keywords
data processing, indexing, data retrieval

Advisors: Slavko Žitnik

Introduction

Search engine indexing is the collecting, parsing, and storing of data to facilitate fast and accurate information retrieval. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in the corpus, which would require considerable time and computing power as we show in section Data retrieval without inverted index. For example, while an index of 10,000 documents can be queried within milliseconds, a sequential scan of every word in 10,000 large documents could take hours. The additional computer storage required to store the index, as well as the considerable increase in the time required for an update to take place, are traded off for the time saved during information retrieval. In this assignment we explore data processing and indexing as well as two approaches for data retrieval:

(A) Data processing with indexing.

(B) Data retrieval with inverted index.

(C) Data retrieval without inverted index.

Implementation

A Data processing with indexing

We implemented the data processing almost entirely as stated in the instructions with the addition of a few stop-words. Through testing and development we noticed that some of the index words were special characters or punctuation's which didn't carry any meaningful value for data retrieval. This is why we decided to exclude such word or tokens.

To keep the project modular and structured we created individual scripts for creating the database, populating the database, preprocessing textual data, printing results and holding entity data as well as making request to the database. We computed the indexes of words in the texts by first splitting the input text by words and then extracting tokens from each individual words. This allowed us to store the correct word indexes rather than token indexes which were in most cases not the same and were presenting a problem when we tried to print the resulting snippets.

In order to have a model layer for better decoupling, we created a separate class for the Posting entity which handled the datatype transformations for the database.

B Data retrieval with inverted index

Data retrieval with inverted index was simple to implement once the index database was populated. We added two arguments to our implementation for ease of use. The first argument was the number of displayed results and the second was the number of displayed snippets per returned result.

The results of data retrieval with inverted index were quick and accurate. Most of the queries were executed and returned results in less than 10ms.

An example of an output with top two results when running run-sqlite-search.py script with the search term "Avtomobil" is shown below:

```
Results for a query: "avtomobil"

Results found in 1ms.
Showing top 1 results limited to 1 snippets.

Frequency 5
Document  podatki.gov.si.200.html
Snippet   ... na osebni avtomobil na leto ...
```

C Data retrieval without inverted index

Data retrieval without inverted index was implemented, as suggested, simply by sequentially opening each file, processing it and than merging the results.

The main takeaway here was that this naive approach is infinitely slower and computation heavy with a large number of files. On average the times to results were around 1min 25sec. The output for the same query as in the previous section is shown below.

```
1 Results for a query: "avtomobil"
2
3 Results found in 86328ms.
4 Showing top 1 results limited to 1 snippets.
5
6 Frequency 5
7 Document  podatki.gov.si.200.html
8 Snippet   ... na osebni avtomobil na leto ...
```

Database

The final populated database contained 48940 index words. Some of these words were longer than any reasonable length and were therefore not useful in data retrieval. This be seen from the Figure 1 which shows the distribution of index word lengths.

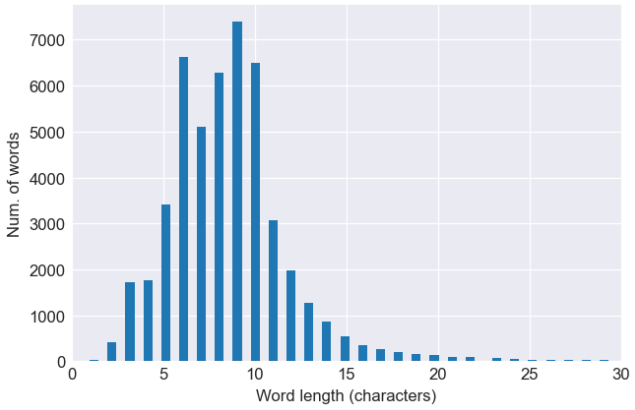

Figure 1. Distribution of index Word length in characters.

We also computed and plotted the combined frequencies of index words in top 20 documents as well as words which are shown in Figure 2 and Figure 3 respectively. The top 3 words with the highest frequency in the top 20 documents with the highest combined frequency are shown in Table 1.

There seem to be one document with a larger amount high frequent words as its sum reaches 80k and the most frequent word seems to be *podatkov* with little over 10k occurrences. This is also a consequence of less than ideal index word extraction where the index words seem to be some special characters, numbers or short character sequences with no real meaning. This is somewhat confirmed in the first row of Table 1, showing this particular document and its most frequent word occurrences, with one of the most frequent words being *'gl'* and *'.'* strings.

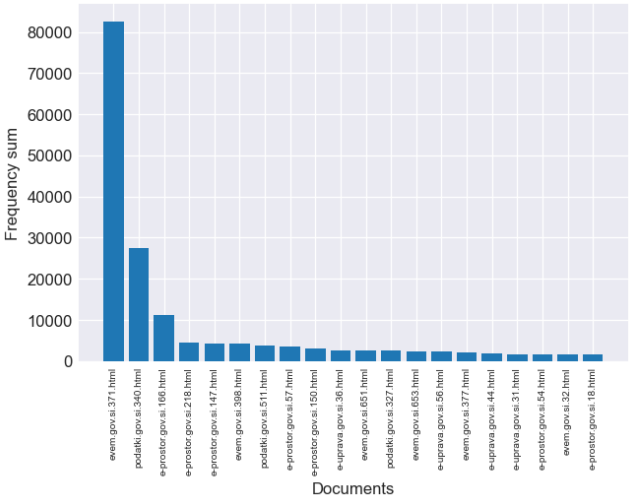

Figure 2. Combined frequency in top 20 documents.

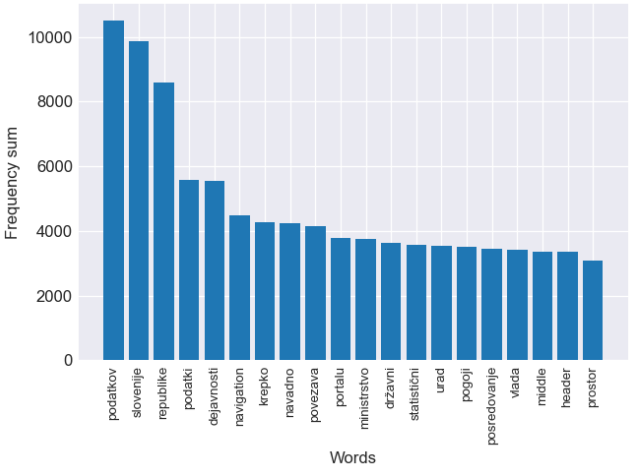

Figure 3. Combined frequency of top 20 words.

The frequencies of individual words are more uniformly distributed than those of documents. They are also more meaningful in terms of telling us the content of the processed text. The top two words *'Podatkov'* and *'Slovenije'* are very descriptive of the content of the official Slovenian government web pages.

Conclusion

In this assignment we focused on data processing and a simulated inverted index storage for further data retrieval. We evaluated the implemented sqlite based search technique with inverted indexing against the basic naive approach with simple sequential document searcher and found that the approach with the inverted index is a lot faster. We analyzed the populated database and pointed out some shortcomings with the extracted index words as well as possible improvements. In conclusion this assignment provided us with a deeper understanding of the inner-workings of search engines and indexing approaches.

Table 1. Frequencies of words grouped by documents and sorted by combined frequency.

document	word	frequency
evem.gov.si/evem.gov.si.371.html	proizvodnja	2266
	gl	1668
	·	1652
podatki.gov.si/podatki.gov.si.340.html	krajevna	754
	skupnost	809
	d.o.o	967
e-prostor.gov.si/e-prostor.gov.si.166.html	om	511
	46	422
	15	320
e-prostor.gov.si/e-prostor.gov.si.218.html	146232	2
	161	2
	380718	2
e-prostor.gov.si/e-prostor.gov.si.147.html	/xsd	582
	simpletype	380
	xsd	925
evem.gov.si/evem.gov.si.398.html	register	42
	ddv	71
	?	80
podatki.gov.si/podatki.gov.si.511.html	gt	464
	lt	464
	quot	284
e-prostor.gov.si/e-prostor.gov.si.57.html	podatke	36
	?	77
	podatkov	51
e-prostor.gov.si/e-prostor.gov.si.150.html	ljubljana	60
	str	53
	geodetski	46
e-uprava.gov.si/e-uprava.gov.si.36.html	območje	176
	1	118
	preverjanje	85
evem.gov.si/evem.gov.si.651.html	drobno	45
	debelo	48
	trgovina	94
podatki.gov.si/podatki.gov.si.327.html	lt	259
	gt	259
	quot	166
evem.gov.si/evem.gov.si.653.html	pooblastilo	61
	dovoljenje	107
	vpis	72
e-uprava.gov.si/e-uprava.gov.si.56.html	otroka	19
	vozila	16
	zveze	17
evem.gov.si/evem.gov.si.377.html	nosilec	69
	obrtne	68
	dejavnosti	75
e-uprava.gov.si/e-uprava.gov.si.44.html	uporabe	27
	euprava	53
	portala	36
e-uprava.gov.si/e-uprava.gov.si.31.html	enota	102
	2019	90
	upravna	102
e-prostor.gov.si/e-prostor.gov.si.54.html	infrastrukture	46
	gospodarske	40
	javne	44