Viktor Petreski and Lodi Dodevska

Report

Third assignment for Web information extraction and retrieval

Introduction

For this assignment, we implemented an inverted index for information retrieval. Then, we queried it to obtain the desired parts of each document. Furthermore, we implemented basic (brute force) search, over the same documents, and compared the execution times.

Implementation

To implement the inverted index successfully, we followed the presented guidelines.

Data processing

The first step in the implementation was the preprocessing of the raw HTML. To extract the contents of the HTML, we used *BeautifulSoup*. Since the content of each page is usually in the *body* tag of the HTML, we extracted only the contents of that tag. Furthermore, we removed unnecessary tags that were part of the body: *script*, *svg* and *footer*. We used the proposed tokenization function (from *nltk*) and extracted the tokens from each text. The next step was removing the stopwords (we also included the English stopwords) from the obtained token list. Since the tokenization did not remove the trailing and leading punctuation marks, we took care of that too. To count the frequency of each token, we used *Counter* from the built-in library *collections*.

Extracting positions of the tokens, was done with a small difference in the processing. Instead of removing the redundant HTML tags from the body, we left them. The removal of the elements proved to be the slowest part of the chain, so eliminating it produced faster snippet generation. Then, this "full" text was processed further by removing the repeating (≥ 2) newlines and whitespaces with one whitespace. After that, we made sure that the text was lowercase and split it on a whitespace. This produced a list of words that represented the content of the HTML (we will refer to this token list with "full" word list). The purpose of this list was to get the true positions of the tokens described in the previous paragraph.

This resulted in two different lists of tokens: one for the words in the index and one for their locations in the "raw" body tag. To find the positions of each of the tokens, we iterated over the full word list and used a dictionary with the word as a key and list of the locations as value. Since this dictionary contained the stopwords that were not needed, when inserting into the database, we extracted only the words contained in the first list of real tokens. For the frequency, we used the values from the Counter dictionary.

Data retrieval

The queries were preprocessed in the same way as the token extraction described previously.

With inverted index

The SQLite queries proved to be limited to a fixed amount of parameters/variables, which was not what we needed. The first solution that we created, was to iterate over the tokens of the query and combine the results in a list. Then we iterated over this list and created a dictionary with the document name as a key. This combined the results from different query words, found in the same document, in one entry. Also, the frequencies were summed and the indexes combined and sorted. This was a good solution to some extent, but not the solution we were looking for.

The next solution that we came up with was to modify the query based on the number of words (x) in the query. This included, adding x number of ? in the query and instead of searching with = (equals), we could search with in operator. With this modification, we could query by a different amount of words with one call to the database and it would take care of concatenation of the results for the different words in the query.

The database grouped the indexes of the words, but it could not sort them since they were added as a string. The next step was to convert them to integers and sort them. This allowed us to find consecutive appearances of the separate words in the query and when this happened, we removed the later appearing index since we did not want the same snippet appearing twice. The processed results were inserted into a dictionary.

To create the snippet, we iterated over the indexes list and extracted ±3 elements around the current index. Also, if the index was at the beginning/end of the document we did not add the dots.

For pretty printing, we used tabulate. The output is also saved to file.

Without inverted index

For the brute force approach, we use a combination of the index creating process and data retrieval with the index.

After obtaining the full list of document locations, we iterate over it and for each document we preprocess the body content to create two arrays: tokens and full word list. If there were matches to the query, we continued. These lists were used in the same manner as described in the preprocessing part, with one difference: instead of inserting the results in the database, we added them to a dictionary. Then, the indexes were sorted, removed consecutive appearances and created a list with the frequency, document and indexes. This list was appended to a main list where all the document matches are added. After iterating over all the documents, this main list was printed using *tabulate*.

Database

We used the proposed structure of the database: IndexWord and Posting table. The columns of the tables were also the same.

Some statistics:

- The algorithm indexed 48555 words
- The most frequent words:

| word | total frequency |
|------------|-----------------|
| slovenije | 9125 |
| podatkov | 8226 |
| republike | 7785 |
| dejavnosti | 4385 |
| krepko | 4277 |

• 3 most frequent documents:

| document | total frequency |
|--|-----------------|
| evem.gov.si/evem.gov.si.371.html | 13111 |
| podatki.gov.si/podatki.gov.si.340.html | 6507 |
| e-prostor.gov.si/e-prostor.gov.si.166.html | 6101 |

• Top 3 most frequent full words in a document (they all come from the same document):

| document | word | frequency |
|-----------------------------------|-------------|-----------|
| evem.gov.si/evem.gov.si.371.html | proizvodnja | 2266 |
| evem.gov.si/evem.gov.si.371.htmll | spada | 1338 |
| evem.gov.si/evem.gov.si.371.html | dejavnosti | 1283 |

Top 5 most distributed words among all documents:

| word | number of unique documents in which the word appears |
|-----------|--|
| domov | 1384 |
| državni | 1238 |
| slovenija | 994 |
| pogoji | 988 |
| področje | 945 |

Results

The queries on the inverted index gave results in a matter of a few milliseconds. Querying the inverted index returns a result within 15 milliseconds¹, which is a huge improvement compared to the classical search which needs more than 90 seconds.

Each row in the tables below shows the sum of frequencies, the document in which the query word(s) were found and the snippets around each query word in that document.

The given query is found in each document in which it appears, regardless of where it is placed (the beginning, middle or end of the document), whether it starts with capital or lowercase letter or if it is completely written with capital letters.

We can see that querying "predelovalne dejavnosti" returns a lot of results. The word "dejavnosti" is quite frequent (4th most frequent) and the huge number of results is expected. For the most frequently indexed document, *evem.gov.si/evem.gov.si.371.html*, this query returns more than 1000 appearances of the keywords. This is only one of the cases where we can see the speediness and the advantage of the inverted index.

On the other hand, the query "social services" returned only 4 document hits, since it is in English and most of the documents are in Slovene.

"Trgovina" appears mostly in two of the most frequently indexed documents: evem.gov.si/evem.gov.si.371.html and podatki.gov.si/podatki.gov.si.340.html. As a matter of fact, more than half of the appearances of "trgovina" are in these two documents.

In our opinion, "ustrezne šifre skladišč" is a good example to demonstrate the performance of the algorithm. We can see that the output contains each subquery of the original that appears in the documents: "ustrezne šifre", "skladišč", "ustrezne".

The two other queries: "agricultural machinery" and "ministry", which are in English, appear fewer times than the other Slovenian queries, just like "social services", but are correctly and fast identified in each document.

¹ note: the program may take a while to process all the snippets and print them out, but the actual queries that we need are return in a few milliseconds

Query results

Please zoom in to see the results. We have limited the number of documents to 19, the number of snippets to 10, after 2-3 snippets we added a new line and each keyword is accompanied by its two neighbours. This configuration is only for these screenshots and the final output that you will get from running both of the scripts is as described above. We tried pasting the raw text here but everything was misaligned so this was the best option. The image is high resolution, so you can zoom in quite a lot without losing the readability. Sorry for the inconvenience!

We have added the full outputs to the README on GitHub for better visualization.

predelovalne dejavnosti

```
Solges

| Top | Property | Proper
```

trgovina

```
Result for purry tryptics
Result from int 11.4 mm | 11.4
```

social services

```
Results for query: social services
Results found in: 10.61 ms
Frequencies Document

S e-uprava_gov.si/e-uprava_gov.si.9.html
S e-uprava_gov.si/e-uprava_gov.si.45.html
Labour, retirement Social services, health, ... relationship etc.7 Social services, health, ... obtain financial social assistance? How ...
Labour, retirement Social services, health, ... relationship etc.7 Social services, health, ... obtain financial social assistance? How ...
Labour, retirement Social services, health, ... relationship etc.7 Social services, health, ... obtain financial social assistance? How ...
Labour, retirement Social services, health, ... relationship etc.7 Social services, health, ... obtain financial social assistance? How ...
Labour, retirement Social services, health, ... relationship etc.7 Social services, health, ... obtain financial social assistance? How ...
Labour, retirement Social services, health, ... relationship etc.7 Social services, health, ... obtain financial social assistance? How ...
Labour, retirement Social services, health, ... relationship etc.7 Social services, health, ... obtain financial social assistance? How ...
Labour, retirement Social services, health, ... relationship etc.7 Social services, health, ... obtain financial social assistance? How ...
Labour, retirement Social services, health, ... relationship etc.7 Social services, health, ... obtain financial social assistance? How ...
Labour, retirement Social services, health, ... relationship etc.7 Social services, health, ... obtain financial social assistance? How ...
```

ustrezne šifre skladišč

agricultural machinery

| Results for qu Results found Frequencies | | Snippet |
|--|---|--|
| 1 1 1 | evem.gov.si/evem.gov.si.651.html e-uprava.gov.si/e-uprava.gov.si.9.html e-uprava.gov.si/e-uprava.gov.si.45.html | maintenance of machinery for general forestry, food Agricultural and rural forestry, food Agricultural and rural |

ministry

| | uery: ministry d in: 11.34 ms | |
|-------------|--|--|
| Frequencies | Document | Snippet |
| 2 | evem.gov.si/evem.gov.si.661.html | following addresses: Ministry of Public 2014, Ministry of Public |
| 1 | e-uprava.gov.si/e-uprava.gov.si.9.html | Privacy protection Ministry of Public |
| 1 | e-uprava.gov.si/e-uprava.gov.si.45.htm | I Privacy protection Ministry of Public |