EDAN20 Assignment 1
Building an inverted index

Matilda Andersson, ma4748an-s

September 2020

1 Introduction

The purpose of this laboratory report is to describe and comment the indexer created in the assignment. A picture of the generated similarity matrix is attached to show the cosine similarities between the different documents based on their tf-idf values. Furthermore, a comparison is made between the indexer constructed in this assignment and Google's indexing algorithms that are presented in the text *Challenges in Building Large-Scale Information Retrieval Systems*.

2 Code breakdown of inverted indexer

2.1 Indexing one file

Breaking down the task into more manageable pieces we started with solving the task of reading and indexing the content of one document. Reading the content of one file, the text was then tokenized using the `finditer()` function in Python's regex library that are filtering out the words from the text using the specified regex encoding `r"\p{L}+"`. The function returned a list of `regex.Match` objects containing the word and its position in the document, which were extracted and placed in a dictionary, with the tokens (words) as keys, and the values being a list of the word's positions.

2.2 Reading the content of a folder

After completing the task of indexing one document, the full content of the `Selma` folder was read, giving us 9 documents where each of them contained the words of a *Selma Lagerlöf* novel. These documents made up the corpus of this assignment.

2.3 Creating the master index

In this section of the assignment the master index were computed. Using the indexing algorithm created for the first sub-task, all the documents in the corpus were indexed and the words present in the corpus were stored as keys in this master index. For each word there was another sub-dictionary associated as value, which in turn contained the name of the document where the word occurred and its positions within this document. This gave us the master index containing the information of all words present in the corpus and were they could be found.

2.4 Using tf-idf representation

The tf-idf representation of a corpus enables us to compare and analyze the different words by giving us a numerical value on how important a word is to a file in the document collection. Using another dictionary, the `tfidf` dictionary, each document is represented with another sub-dictionary. The keys in the `tfidf` dictionary is the document name and its value is a sub-dictionary were, in turn, its keys being all the words that are present in the corpus and as corresponding value, the tf-idf metric of the word in that document. Tf, in this assignment, is the relative frequency of the term in the document and, idf, the logarithm base 10 of the inverse document frequency. The lower the value (with 0 being the lowest) the less relevant the word is to the document.

2.5 Comparing the documents

For the last task, we were to compare the similarities of all the documents in the corpus by calculating the cosine similarity based on the documents tf-idf representation. The resulting similarity matrix is presented in fig.1, where the files `herrgard.txt` and `jerusalem.txt` having the highest similarity score of 0.3706894238733847.

```
['bannlyst', 'gosta', 'herrgard', 'jerusalem', 'kejsaren', 'marbacka', 'nils', 'osynliga', 'troll']

Out[40]: array([[0.        , 0.04903692, 0.00094869, 0.00645979, 0.02400866,
                 0.03681049, 0.05098228, 0.05205524, 0.08862113],
                [0.04903692, 0.        , 0.0031104 , 0.00432096, 0.04801793,
                 0.08016786, 0.10482617, 0.12475544, 0.1957378 ],
                [0.00094869, 0.0031104 , 0.        , 0.37068942, 0.00073955,
                 0.00361428, 0.00506797, 0.00482553, 0.00407382],
                [0.00645979, 0.00432096, 0.37068942, 0.        , 0.00183431,
                 0.00487261, 0.00453913, 0.02829997, 0.0070571 ],
                [0.02400866, 0.04801793, 0.00073955, 0.00183431, 0.        ,
                 0.07112071, 0.049663  , 0.05110844, 0.18128404],
                [0.03681049, 0.08016786, 0.00361428, 0.00487261, 0.07112071,
                 0.        , 0.0847413 , 0.09316826, 0.14715377],
                [0.05098228, 0.10482617, 0.00506797, 0.00453913, 0.049663  ,
                 0.0847413 , 0.        , 0.11057352, 0.18847519],
                [0.05205524, 0.12475544, 0.00482553, 0.02829997, 0.05110844,
                 0.09316826, 0.11057352, 0.        , 0.19259654],
                [0.08862113, 0.1957378 , 0.00407382, 0.0070571 , 0.18128404,
                 0.14715377, 0.18847519, 0.19259654, 0.        ]])
```

Figure 1: Table showing the cosine similarity of the documents in the corpus.

3 Google's indexing technique

Reading through the slides describing the indexing technique used by Google throughout the years, one slide stood out to me, which was **slide number 45**, which describes Google's indexing technique being based on a website's docid and the number of hits it had. An approach which is similar to the term frequency we used in this assignment.