EDAN20 - Lab 1

André Frisk - an8218fr-s

October 11, 2021

The Indexer

This indexer which the students were tasked to write was split into different parts which would merge together at the end. This lab is about reading the contents of a file/folder and calculate the master index and TF-IDF (explained later) which will be used to compare documents how similar they are. The books are written by Selma Lagerlöf. The tokenization regex used in this lab was formed such as:

```
regex = r"\p{L}+"
```

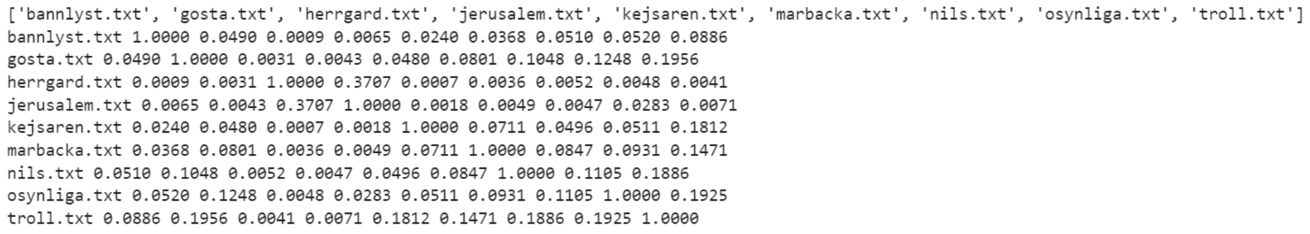
Firstly, a function is created which takes two inputs, the name of a directory and the suffix (in this case .txt). This function traverses the directory and checks if a file ends with this suffix and insert them into a list which is then returned to the user. This function is used for every function further on in the lab as the books (text files) is required.

Master Index

The master index is a dictionary where every word is stored in the Selma-folder and specifies in which of the texts this word is involved in and gives out the position of where the word is in the text. The previous functions *tokenize()* and *text_to_idx()*, is used here to tokenize the text and point out the positions of the different words one file at a time. Then we just insert the word and the file with the positions in the dictionary based if it's already there or not. As the master index is a dictionary, a simple call with ex. *master_index['samlar']* will print out the dictionary for the word *samlar* which prints out every file the word occurs in and in which index the words position is in the text.

Term Frequency - Inverse Document Frequency

The texts will also be represented by a TF-IDF dictionary. TF-IDF is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. TF stands for *Term Frequency* and is calculated with the formula: $tf = \frac{f(t,d)}{N(d)}$ where $f(t,d)$ is the number of times f occurs in the document d and $N(d)$ is the total amount of words in the document. IDF is *Inverse Document Frequency* which is calculated with the formula: $IDF = log\frac{N}{d_t}$ where N is the total documents in the collection or corpus and d_t is the number of documents that the term t appears in. The TF and IDF is then multiplied with each other to give out the TF-IDF for that specific word. These values are inserted in a dictionary where every file have a TF-IDF value for a specific word.



```
['bannlyst.txt', 'gosta.txt', 'herrgard.txt', 'jerusalem.txt', 'kejsaren.txt', 'marbacka.txt', 'nils.txt', 'osynliga.txt', 'troll.txt']
bannlyst.txt 1.0000 0.0490 0.0009 0.0065 0.0240 0.0368 0.0510 0.0520 0.0886
gosta.txt 0.0490 1.0000 0.0031 0.0043 0.0480 0.0801 0.1048 0.1248 0.1956
herrgard.txt 0.0009 0.0031 1.0000 0.3707 0.0007 0.0036 0.0052 0.0048 0.0041
jerusalem.txt 0.0065 0.0043 0.3707 1.0000 0.0018 0.0049 0.0047 0.0283 0.0071
kejsaren.txt 0.0240 0.0480 0.0007 0.0018 1.0000 0.0711 0.0496 0.0511 0.1812
marbacka.txt 0.0368 0.0801 0.0036 0.0049 0.0711 1.0000 0.0847 0.0931 0.1471
nils.txt 0.0510 0.1048 0.0052 0.0047 0.0496 0.0847 1.0000 0.1105 0.1886
osynliga.txt 0.0520 0.1248 0.0048 0.0283 0.0511 0.0931 0.1105 1.0000 0.1925
troll.txt 0.0886 0.1956 0.0041 0.0071 0.1812 0.1471 0.1886 0.1925 1.0000
```

Figure 1: The similarity for each text compared to the other texts

Comparing documents

To compare the documents a function *cosine_similarity(document1, document2)* is created to point out how similar the two documents are. This is used in the similarity matrix to show how similar the documents are in the Selma corpus. Here the code compares one file with all the others with the cosine similarity function and insert it in a matrix. Also two important variables *most_sim_doc1* and *most_sim_doc2* is focused here as if two documents are more similar than the current maximum similarity these two files become the new *most_sim_docs*. When the matrix is done and all files have been checked the matrix is printed out 1, also the most similar documents are printed out. In this lab it was *herrgard.txt* and *jerusalem.txt* with a similarity of 0.3706978453750167.

Google Indexing

The lab apply indexing in matrix dictionaries. Google use alot of indexing and their servers contain huge amount of information for these indexes. However, what we did in this lab doesn't require a server as its quite a small scale but the idea is there. What is done here is trying to improve the response time of receiving the information and try to have a low latency. The matrix (compare documents or master index) is quite efficient in this point of view as the user can quite fast get information about a specific text or compare documents, even retrieve them is fast.

Most similar indexing technique: slide number 45, "Index Encoding circa 1997-1999".