CMSC 676
Information Retrieval
Project-III

Bhushan Mahajan
m302@umbc.edu

Project Overview:

- This project is built on previous project's functionalities, i.e. document-preprocessing from corpus of HTML files and calculating TF-IDF of a token in a document.
- In this project, I have used hashmap in Python which is defaultdict from collections library. Lookup time of hashmap is O(1).
- After all the calculations are done, I created the Term Document Matrix (TDM) using defaultdict which maps a token to a list in which document-id and token's tf-idf.
- Dictionary file is also created with the help of defaultdict.

Development Environment:

- Processor: Apple Silicon M1 (ARM-64)
- Programming Language- Python-3.8
- Text Editor: VS Code

Project Execution:

1. Go to the project directory
2. Run the following command:
    - *python3 inverted-index.py <input-files-dir-path> <output-file-dir-path>*

Inverted Indexing Process Explanation:

- Pre-processing:
    - Preprocessing is the first step to calculate inverted indices. I have used pre-processing functionality from Project-1 and Project-2 in which I extracted tokens using "Gensim" tokenizer.
    - Furthermore, I removed stop-words, words having length 1, words that occur only once in entire corpus. Also, I removed punctuations, numbers, white spaces, etc.

- TF-IDF calculation:
  - Terminology:
    - t: term/token
    - d: document (set of tokens)
    - N: count of corpus (For this project, N = 503)
    - corpus: the total document set
    - tf: term frequency
    - df: count of documents in which token t occurs
    - idf: inverse document frequency
    - tf-idf(t,d): weights of a token t in document d.

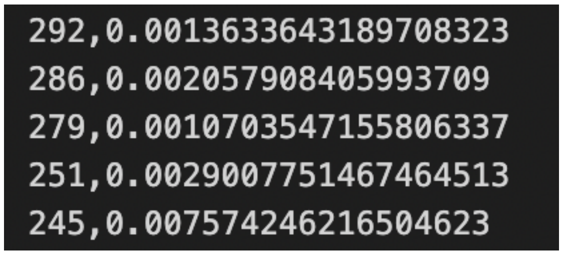
  - To calculate TF-IDF, I used the formula: ***tf-idf(t,d) = tf(t,d) * idf(t)***, Where,

    tf(t,d) = occurrence of t in d / total numbers of tokens in d.

    idf(t) = $\boldsymbol{\log_e(N/df(t))}$
  - Since we have large corpus, to avoid explosion of *idf* value, I have taken *natural log* of *idf* to dampen the effect.
- Inverted index processing:
  - An inverted index is a data structure that stores a mapping between information, such as words or integers, and their places in a document or group of documents. In layman's terms, it's a hashmap-like data structure that guides you from a word to a text or a web page.
  - Position-hashmap:
    - This is a hashmap that contains array of hashmap as a value for a given token as key.
      i.e. MAP(Token, LIST(MAP(document_number, tf-idf)))
    - This is a global hashmap and values are inserted right after tf-idf calculation is done.
  - Dictionary-hashmap:
    - This is a hashmap that maps the token to another hashmap of number of documents in which that token occurs and document number of first occurrence of the token.
    - i.e MAP(Token, LIST(MAP(Number of documents, Document of first occurrence)))

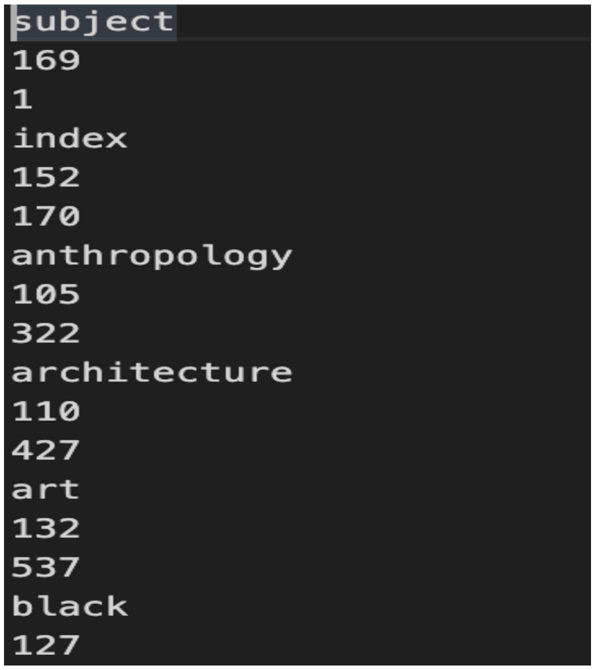
Output Sample:

- The postings output file contains document number in which a token occurs and TF-IDF of the token. Below is the sample output:



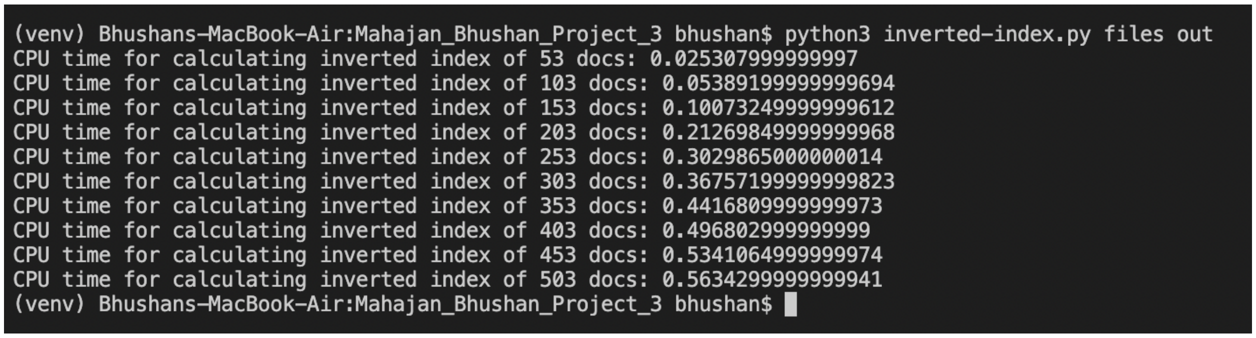
```
292,0.0013633643189708323
286,0.002057908405993709
279,0.0010703547155806337
251,0.0029007751467464513
245,0.007574246216504623
```

- The dictionary file contains token, the number of documents in which the token occurs, and document number in which first occurrence is detected.



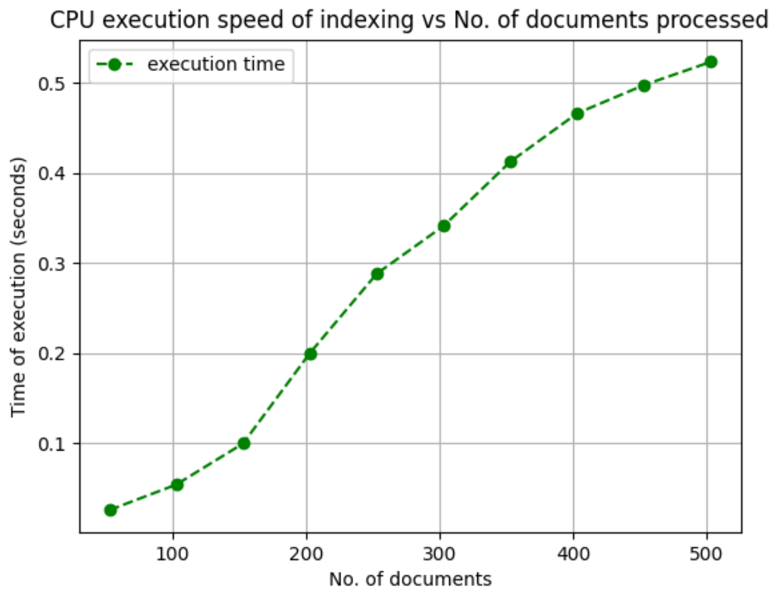
```
subject
169
1
index
152
170
anthropology
105
322
architecture
110
427
art
132
537
black
127
```

- I have calculated the CPU execution time of the program. Time recording starts from pre-processing function and recording is stopped when posting file and dictionary file is written. Below is the execution time output:



```
(venv) Bhushans-MacBook-Air:Mahajan_Bhushan_Project_3 bhushan$ python3 inverted-index.py files out
CPU time for calculating inverted index of 53 docs: 0.025307999999997
CPU time for calculating inverted index of 103 docs: 0.05389199999999694
CPU time for calculating inverted index of 153 docs: 0.10073249999999612
CPU time for calculating inverted index of 203 docs: 0.21269849999999968
CPU time for calculating inverted index of 253 docs: 0.3029865000000014
CPU time for calculating inverted index of 303 docs: 0.36757199999999823
CPU time for calculating inverted index of 353 docs: 0.4416809999999973
CPU time for calculating inverted index of 403 docs: 0.496802999999999
CPU time for calculating inverted index of 453 docs: 0.5341064999999974
CPU time for calculating inverted index of 503 docs: 0.5634299999999941
(venv) Bhushans-MacBook-Air:Mahajan_Bhushan_Project_3 bhushan$
```

- Below is the graph of the indexing time as a function of the number of documents in your report



How the total size of the output files, dictionary+postings, compares to the size of the input files, for various numbers of input files?

- We can clearly observe that both the output files i.e posting and dictionary files have compressed information of entire corpus of 503 raw HTML files, which results in less size.