

1: The steps I have performed for data preprocessing and indexing.

`preprocess_and_index.py` file handles data processing and indexing. I read the smg files parallelly using threads. Each thread read the file article by article. Since each article is surrounded with <REUTERS> tag, I used the substring and find() methods of strings. After storing the article in a variable, I extracted the “NEWID”, <TITLE> and <BODY> parts using the same methodology. If none were found, I assumed it was an empty string.

After merging the title and the body, I did lowercase every character with “lower()” method, removed “\n”s, removed stopwords with “replace()” method, removed punctuations using “translate()” method and “string.punctuation” list, and finally removed stopwords again. I removed stopwords twice, because I realized that some were still there after the first removal. I also removed numbers. So, my cosine similarity scores will be higher than the others’ if they did not remove numeric terms.

Then, I started positional indexing. I used a bottom up approach: started with words, then articles, then sgml files and finally merged everything.

First, I inverted articles in the `invertArticle(ID, TEXT)` function. For every word in the article, I stored the locations of the word in the article. Final form of the inverted article looks like this: {"term_1": {ID: [i1, i2, i3, ...]}, "term_2": {ID: [i1, i2, i3, ...]}, "term_3": {ID: [i1, i2, i3, ...]}, "term_4": {ID: [i1, i2, i3, ...]}, ...}. ID is the id of the document, “term_x”s are words and i values are locations(indexes) of the words in the article.

Secondly, I merged articles of a file in the `merge(postings)` function. Final form of the merged articles is like this: {"term_1": {ID1: [i1, i2, i3, ...], ID2: [i1, i2, i3, ...], ID999: [i1, i2, i3, ...]}, "term_2": {ID13: [i1, i2, i3, ...], ID2242: [i1, i2, i3, ...], ID10919: [i1, i2, i3, ...]}, "term_3": {ID241: [i1, i2, i3, ...], ID1221: [i1, i2, i3, ...], ID14999: [i1, i2, i3, ...]}, ...}. We can interpret this dictionary variable as follows: term3 is in document-241, document-1221 and document-14999 located in i1, i2, i3, ... indexes.

Finally, I merged these dicts into one in the final lines of the `main()` function by simply appending starting from the first file to the last in ascending order. As a result, the final index became automatically sorted without any extra sorting mechanism.

As the final step, I dumped this index to a json file named “inverted_index.json”. I did neither indent nor sorted the terms since it was meant to be read by another python program `--query_eval.py--`, not a human. This file has a size of 22MB.

I created another index for term frequencies. I used the bag of words approach. For every document, I stored the terms and their log scaled frequencies in the index. Final form of the index is as follows: {"ID1": {"term1": TF1, "term2": TF2, "term3": TF3, ...}, "ID2": {"term1": TF1, "term2": TF2, "term3": TF3, ...}, ...}. This index was much needed since I

needed to calculate TF-IDF scores of documents. This index is stored in the “document_frequency_index.json” file. This file has a size of 23.9 MB. Considering all Reuters dataset is 27.9 MB, my indexes occupy 46 MB disk space. So, I must admit my solution is not space efficient, but runtime is very fast.

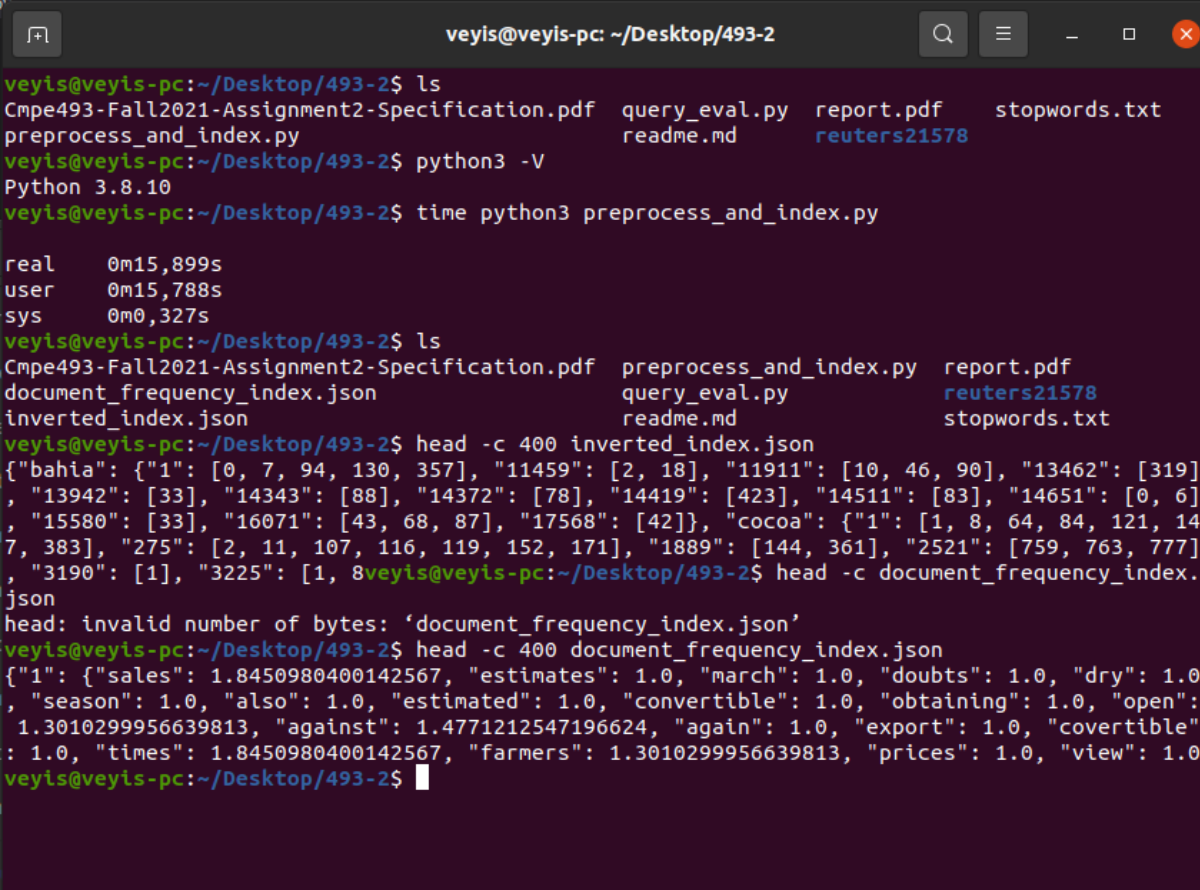
2: The data structures that I used for representing the inverted index

I used python’s built-in “dictionary” collection for representing both the inverted index and the document-term frequency index. Key values are terms --lowercased and stopwords removed-- and values are the corresponding position lists with document ids for inverted index. And for document-term frequency index, keys are document ids and values are the dict of terms-TF values.

I believe this approach can be extended with new articles easily. We will just append the new ID’s to the end of the current positional lists. But if we were to change an existing article, it would be costly since term frequencies must be calculated again and some of the locations of terms might need to be deleted.

The indexes are constructed approximately within 15 seconds on my computer.

3: A screenshot of running the indexing module of the system.



```
veyis@veyis-pc: ~/Desktop/493-2
veyis@veyis-pc:~/Desktop/493-2$ ls
Cmpe493-Fall2021-Assignment2-Specification.pdf  query_eval.py  report.pdf  stopwords.txt
preprocess_and_index.py                       readme.md      reuters21578
veyis@veyis-pc:~/Desktop/493-2$ python3 -V
Python 3.8.10
veyis@veyis-pc:~/Desktop/493-2$ time python3 preprocess_and_index.py

real    0m15,899s
user    0m15,788s
sys     0m0,327s
veyis@veyis-pc:~/Desktop/493-2$ ls
Cmpe493-Fall2021-Assignment2-Specification.pdf  preprocess_and_index.py  report.pdf
document_frequency_index.json                  query_eval.py            reuters21578
inverted_index.json                           readme.md                stopwords.txt
veyis@veyis-pc:~/Desktop/493-2$ head -c 400 inverted_index.json
{"bahia": {"1": [0, 7, 94, 130, 357], "11459": [2, 18], "11911": [10, 46, 90], "13462": [319],
, "13942": [33], "14343": [88], "14372": [78], "14419": [423], "14511": [83], "14651": [0, 6],
, "15580": [33], "16071": [43, 68, 87], "17568": [42]}, "cocoa": {"1": [1, 8, 64, 84, 121, 14
7, 383], "275": [2, 11, 107, 116, 119, 152, 171], "1889": [144, 361], "2521": [759, 763, 777]
, "3190": [1], "3225": [1, 8veyis@veyis-pc:~/Desktop/493-2$ head -c document_frequency_index.
json
head: invalid number of bytes: 'document_frequency_index.json'
veyis@veyis-pc:~/Desktop/493-2$ head -c 400 document_frequency_index.json
{"1": {"sales": 1.8450980400142567, "estimates": 1.0, "march": 1.0, "doubts": 1.0, "dry": 1.0
, "season": 1.0, "also": 1.0, "estimated": 1.0, "convertible": 1.0, "obtaining": 1.0, "open":
1.3010299956639813, "against": 1.4771212547196624, "again": 1.0, "export": 1.0, "convertible"
: 1.0, "times": 1.8450980400142567, "farmers": 1.3010299956639813, "prices": 1.0, "view": 1.0
veyis@veyis-pc:~/Desktop/493-2$
```

4: Screenshots of running the system for each of the two types of queries.

4.1: phrase

```
veyis@veyis-pc:~/Desktop/493-2$ python3 query_eval.py
What is your search query? (Press q for quit.) "old crop cocoa"
Result is saved to result_2021-12-02 15:01:28.236684.json file. You can see it after closing the program.
What is your search query? (Press q for quit.) "turkish economy"
Result is saved to result_2021-12-02 15:01:28.236684.json file. You can see it after closing the program.
What is your search query? (Press q for quit.) q
Now you can see all the contents in result_2021-12-02 15:01:28.236684.json file.
veyis@veyis-pc:~/Desktop/493-2$ cat 'result_2021-12-02 15:01:28.236684.json'
{
  "old crop cocoa": {
    "1": [
      [
        82,
        83,
        84
      ]
    ]
  },
  "turkish economy": {
    "1611": [
      [
        322,
        323
      ],
      [
        356,
        357
      ]
    ],
    "7105": [
      [
        35,
        36
      ],
      [
        161,
        162
      ]
    ]
  }
}
```

Note: At first, I printed out the exact locations of the phrases too. But, I realized that you only want document ids. So, I changed that to printing only document ids.

```
veyis@veyis-pc: ~/Desktop/493-2
veyis@veyis-pc:~/Desktop/493-2$ ls
Cmpe493-Fall2021-Assignment2-Specification.pdf  preprocess_and_index.py  report.pdf
document_frequency_index.json                  query_eval.py           reuters21578
inverted_index.json                            readme.md               stopwords.txt
veyis@veyis-pc:~/Desktop/493-2$ python3 query_eval.py
What is your search query? (Press q for quit.) "old crop cocoa"
Result is saved to result_2021-12-07 21:54:25.626083.json file. You can see it after closing
the program.
What is your search query? (Press q for quit.) "turkish economy"
Result is saved to result_2021-12-07 21:54:25.626083.json file. You can see it after closing
the program.
What is your search query? (Press q for quit.) q
Now you can see all the contents in result_2021-12-07 21:54:25.626083.json file.
veyis@veyis-pc:~/Desktop/493-2$ cat 'result_2021-12-07 21:54:25.626083.json'
{
  "old crop cocoa": [
    1
  ],
  "turkish economy": [
    1611,
    7105
  ]
}
```

```
veyis@veyis-pc: ~/Desktop/493-2
veyis@veyis-pc:~/Desktop/493-2$ ls
Cmpe493-Fall2021-Assignment2-Specification.pdf  readme.md
document_frequency_index.json                  report.pdf
inverted_index.json                            'result_2021-12-07 21:54:25.626083.json'
preprocess_and_index.py                       reuters21578
query_eval.py                                 stopwords.txt
veyis@veyis-pc:~/Desktop/493-2$ python3 query_eval.py
What is your search query? (Press q for quit.) "industrial automation"
Result is saved to result_2021-12-07 21:55:25.694200.json file. You can see it after closing
the program.
What is your search query? (Press q for quit.) "saddam hussein"
Result is saved to result_2021-12-07 21:55:25.694200.json file. You can see it after closing
the program.
What is your search query? (Press q for quit.) "economic troubles"
Result is saved to result_2021-12-07 21:55:25.694200.json file. You can see it after closing
the program.
What is your search query? (Press q for quit.) q
Now you can see all the contents in result_2021-12-07 21:55:25.694200.json file.
veyis@veyis-pc:~/Desktop/493-2$ cat 'result_2021-12-07 21:55:25.694200.json'
{
  "industrial automation": [
    1414,
    10473,
    16600
  ],
  "saddam hussein": [
    7350,
    8093,
    8109,
    8188,
    8405,
    8493,
    13527
  ],
  "economic troubles": [
    8796,
    10733,
    10799
  ]
}
```

```
veyis@veyis-pc: ~/Desktop/493-2
veyis@veyis-pc:~/Desktop/493-2$ ls
Cmpe493-Fall2021-Assignment2-Specification.pdf  report.pdf
document_frequency_index.json                 'result_2021-12-07 21:54:25.626083.json'
inverted_index.json                           'result_2021-12-07 21:55:25.694200.json'
preprocess_and_index.py                       reuters21578
query_eval.py                                 stopwords.txt
readme.md

veyis@veyis-pc:~/Desktop/493-2$ python3 query_eval.py
What is your search query? (Press q for quit.) "proforma"
Result is saved to result_2021-12-07 21:57:03.558639.json file. You can see it after closing the program.
What is your search query? (Press q for quit.) "usage"
Result is saved to result_2021-12-07 21:57:03.558639.json file. You can see it after closing the program.
What is your search query? (Press q for quit.) ""
Wrong input, please try again!
What is your search query? (Press q for quit.) "
Wrong input, please try again!
What is your search query? (Press q for quit.) "dosage"
Wrong input, please try again!
What is your search query? (Press q for quit.) q
Now you can see all the contents in result_2021-12-07 21:57:03.558639.json file.
veyis@veyis-pc:~/Desktop/493-2$ cat 'result_2021-12-07 21:57:03.558639.json'
{
  "proforma": [
    331,
    11925,
    12136,
    13263
  ],
  "usage": [
    327,
    853,
    1041,
    1054,
    4951,
    5827,
    6259,
    6939,
    8824,
    9058,
    9537,
    9892,
    10200,
    11953,
    11961,
    12126,
    12709,
    12720,
    12971,
    14313,
    15716,
    15853,
    15875,
```

4.2: free text

```
veyis@veyis-pc: ~/Desktop/493-2
veyis@veyis-pc:~/Desktop/493-2$ ls
Cmpe493-Fall2021-Assignment2-Specifcation.pdf  report.pdf
document_frequency_index.json                 'result_2021-12-07 21:54:25.626083.json'
inverted_index.json                           'result_2021-12-07 21:55:25.694200.json'
preprocess_and_index.py                       'result_2021-12-07 21:57:03.558639.json'
query_eval.py                                reuters21578
readme.md                                    stopwords.txt
veyis@veyis-pc:~/Desktop/493-2$ python3 query_eval.py
What is your search query? (Press q for quit.) turkish economy
Result is saved to result_2021-12-07 22:05:11.708770.json file. You can see it after closing the program.
What is your search query? (Press q for quit.) q
Now you can see all the contents in result_2021-12-07 22:05:11.708770.json file.
veyis@veyis-pc:~/Desktop/493-2$ python3 query_eval.py
What is your search query? (Press q for quit.) turkish economy economy economy
Result is saved to result_2021-12-07 22:05:18.717706.json file. You can see it after closing the program.
What is your search query? (Press q for quit.) q
Now you can see all the contents in result_2021-12-07 22:05:18.717706.json file.
veyis@veyis-pc:~/Desktop/493-2$ head -n 14 'result_2021-12-07 22:05:11.708770.json'
{
  "turkish economy": [
    [
      "444",
      0.3531674131821311
    ],
    [
      "11944",
      0.32510953310829344
    ],
    [
      "3076",
      0.3249730906087384
    ]
  ],
  "turkish economy economy economy": [
    [
      "444",
      0.3126227254451887
    ],
    [
      "11944",
      0.28778597490848545
    ],
    [
      "3076",
      0.2876651964207618
    ]
  ]
}
veyis@veyis-pc:~/Desktop/493-2$ REALIZE THAT COSINE SCORES ARE DIFFERENT
```

```
veyis@veyis-pc: ~/Desktop/493-2
veyis@veyis-pc:~/Desktop/493-2$ ls
Cmpe493-Fall2021-Assignment2-Specification.pdf  'result_2021-12-07 21:54:25.626083.json'
document_frequency_index.json                  'result_2021-12-07 21:55:25.694200.json'
inverted_index.json                            'result_2021-12-07 21:57:03.558639.json'
preprocess_and_index.py                       'result_2021-12-07 22:05:11.708770.json'
query_eval.py                                 'result_2021-12-07 22:05:18.717706.json'
readme.md                                     reuters21578
report.pdf                                     stopwords.txt
veyis@veyis-pc:~/Desktop/493-2$ python3 query_eval.py
What is your search query? (Press q for quit.) cocoa export shipment tonne
Result is saved to result_2021-12-07 22:11:16.721031.json file. You can see it after closing the program.
What is your search query? (Press q for quit.) q
Now you can see all the contents in result_2021-12-07 22:11:16.721031.json file.
veyis@veyis-pc:~/Desktop/493-2$ head -n 29 'result_2021-12-07 22:11:16.721031.json'
{
  "cocoa export shipment tonne": [
    [
      "1394",
      0.34141761398188325
    ],
    [
      "12348",
      0.2815093079594149
    ],
    [
      "10491",
      0.26288530836186
    ],
    [
      "10471",
      0.2536817687819774
    ],
    [
      "3190",
      0.25360503098308534
    ],
    [
      "19358",
      0.2510514144148947
    ],
    [
      "14721",
      0.249855933395393
    ]
  ]
}
veyis@veyis-pc:~/Desktop/493-2$
```



```
result_2021-12-07 22:13:07.115836.json
veyis@veyis-pc:~/Desktop/493-2$ python3 query_eval.py
What is your search query? (Press q for quit.) nonconvertible profiles
Result is saved to result_2021-12-07 22:13:07.115836.json file. You can see it after closing the program.
What is your search query? (Press q for quit.) q
Now you can see all the contents in result_2021-12-07 22:13:07.115836.json file.
veyis@veyis-pc:~/Desktop/493-2$ cat 'result_2021-12-07 22:13:07.115836.json'
{
  "nonconvertible profiles": [
    [
      "9506",
      0.20765524850219397
    ],
    [
      "6770",
      0.12048993414580213
    ],
    [
      "1553",
      0.09931223957539903
    ],
    [
      "11831",
      0.0974474737398553
    ],
    [
      "10406",
      0.09255904378942752
    ],
    [
      "4007",
      0.09143975506168597
    ],
    [
      "10375",
      0.06456814135929796
    ],
    [
      "319",
      0.0632737430550131
    ],
    [
      "10268",
      0.056796997292467954
    ],
    [
      "15488",
      0.05543415830225518
    ]
  ]
}
veyis@veyis-pc:~/Desktop/493-2$
```

```
veyis@veyis-pc: ~/Desktop/493-2
veyis@veyis-pc:~/Desktop/493-2$ python3 query_eval.py
What is your search query? (Press q for quit.) prescription rheumatoid labs
Result is saved to result_2021-12-07 22:15:52.743887.json file. You can see it after closing the program.
What is your search query? (Press q for quit.)
Wrong input, please try again!
What is your search query? (Press q for quit.) breakeven lawsuit liquidating
Result is saved to result_2021-12-07 22:15:52.743887.json file. You can see it after closing the program.
What is your search query? (Press q for quit.) q
Now you can see all the contents in result_2021-12-07 22:15:52.743887.json file.
veyis@veyis-pc:~/Desktop/493-2$ head -n 30 'result_2021-12-07 22:15:52.743887.json'
[
  "prescription rheumatoid labs": [
    [
      "321",
      0.2414082914565476
    ],
    [
      "7843",
      0.20211377742804051
    ],
    [
      "16259",
      0.18618575678830668
    ],
    [
      "1635",
      0.16560955045126263
    ],
    [
      "8562",
      0.16386465557918906
    ],
    [
      "2933",
      0.15115711799109285
    ],
    [
      "7935",
      0.14357377998827273
    ]
  ],
  ]
}
```