Adalet Veyis Turgut
2017400210
20/11/2021

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

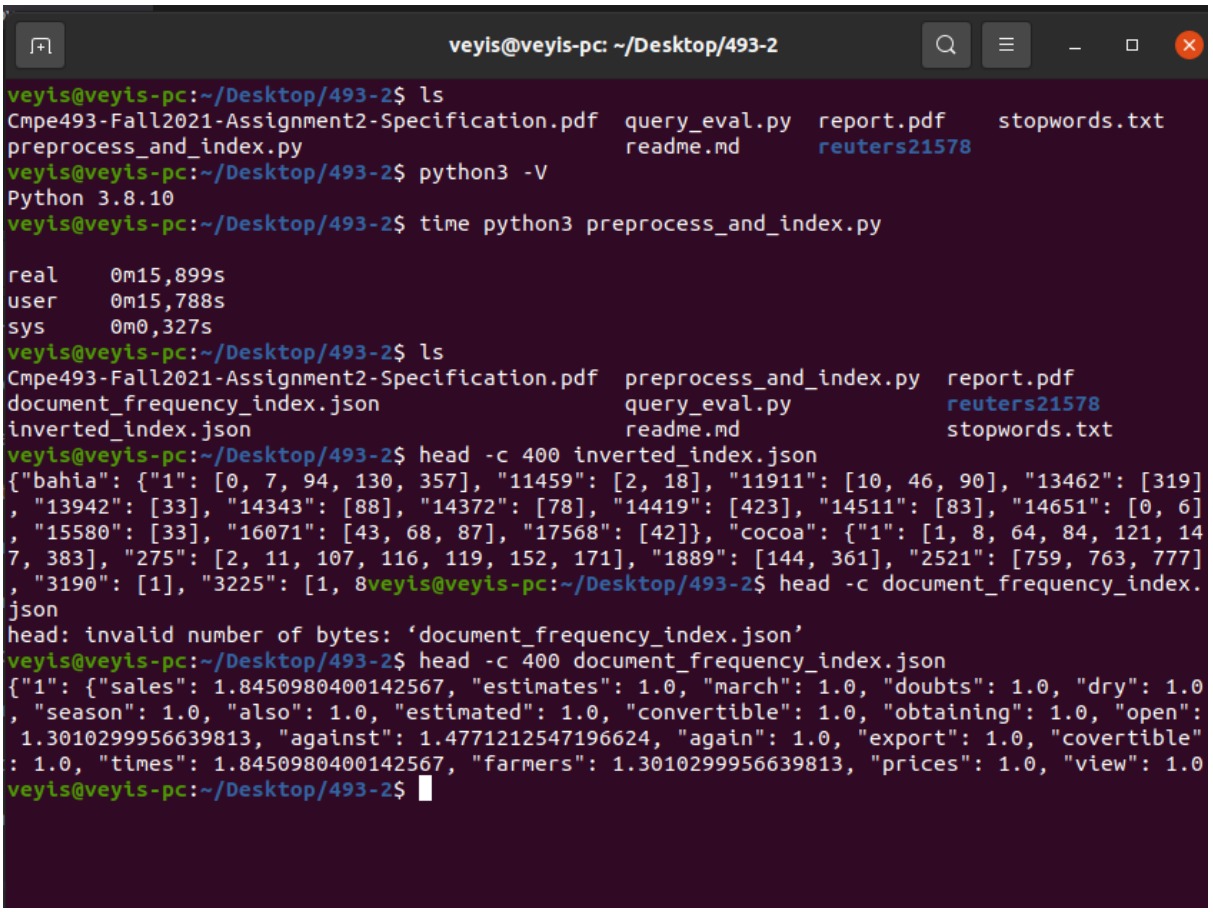
I used python's built-in "dictionary" collection for representing both the inverted index and the document-term frequency index. Key values are terms --lowercased and stopwords removed-- and values are the corresponding position lists with document ids for inverted index. And for document-term frequency 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.

Adalet Veyis Turgut
2017400210
20/11/2021

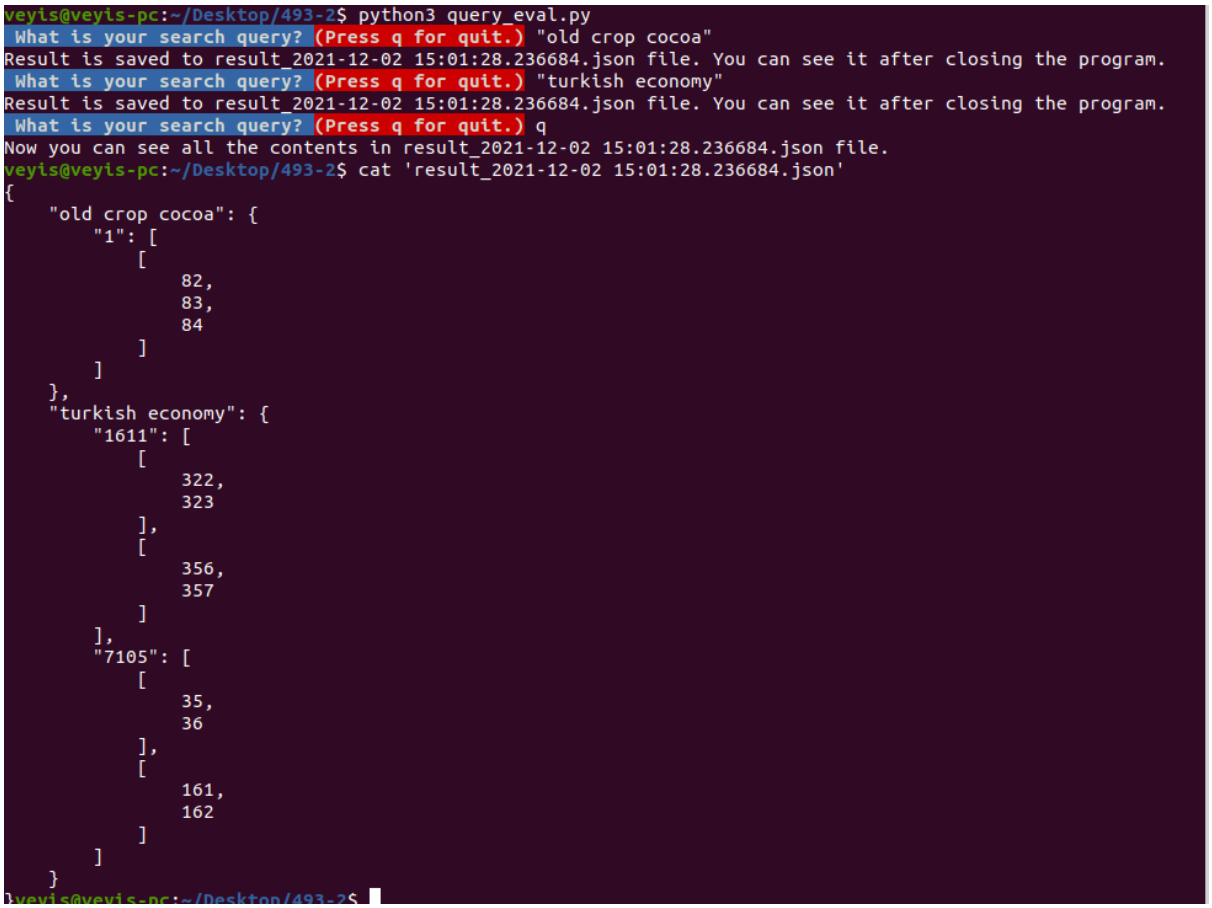
3: A screenshot of running the indexing module of the system.



Adalet Veyis Turgut
2017400210
20/11/2021

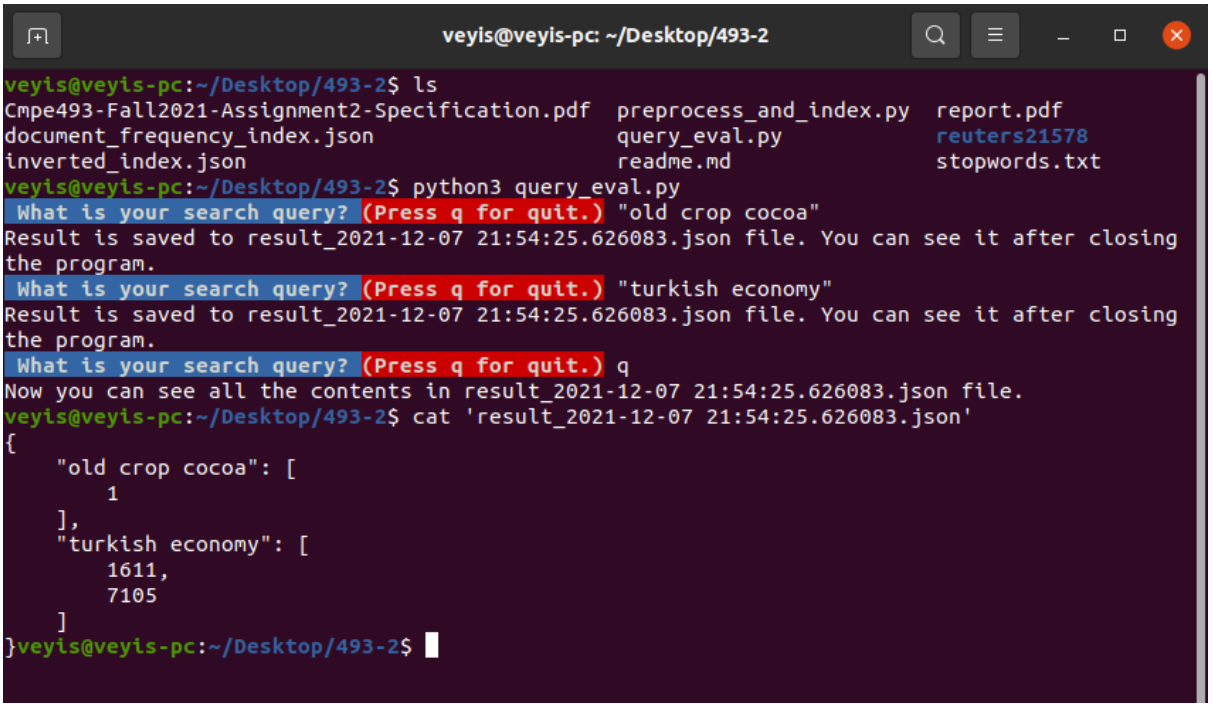
4: Screenshots of running the system for each of the two types of queries.

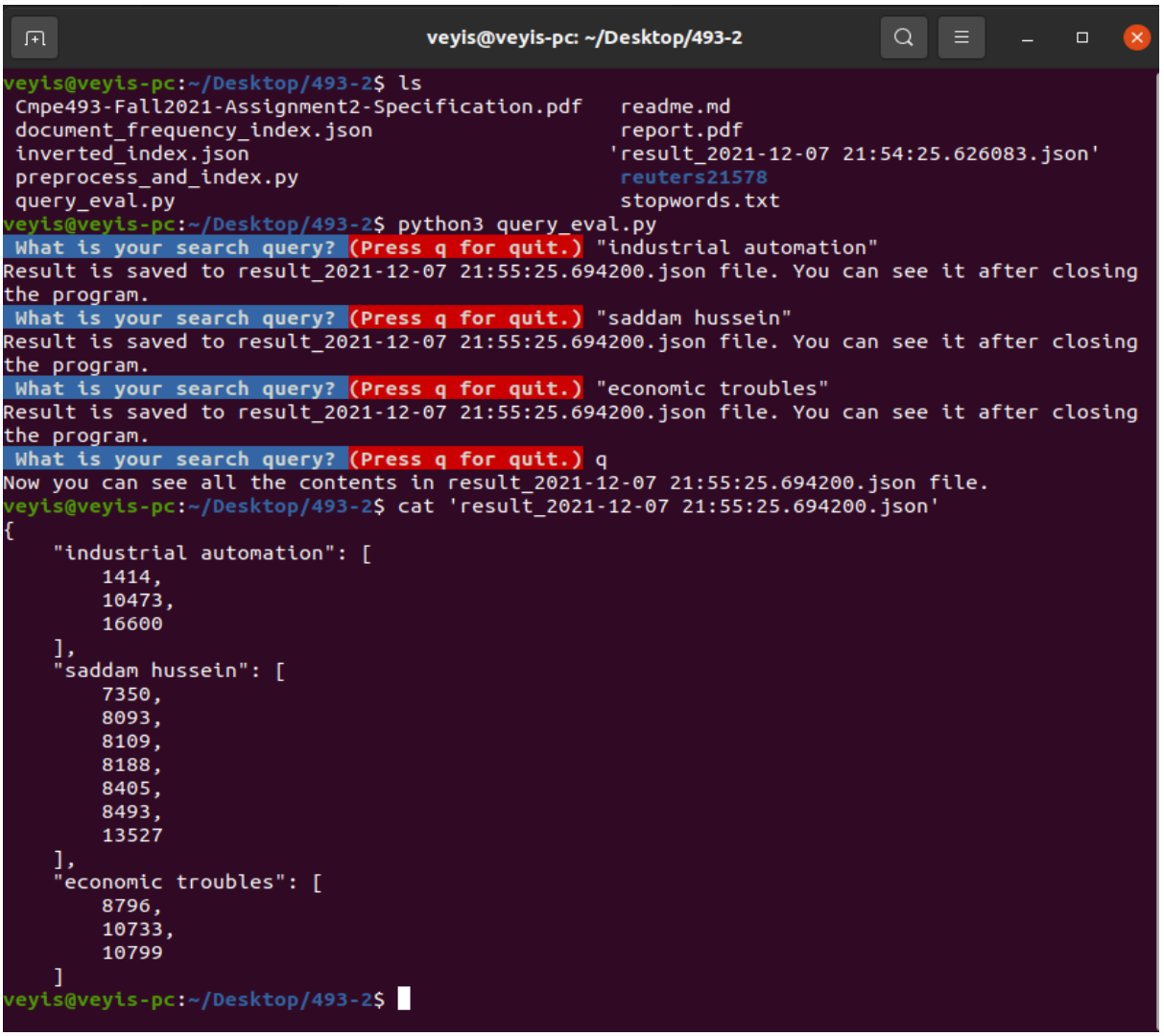
4.1: phrase



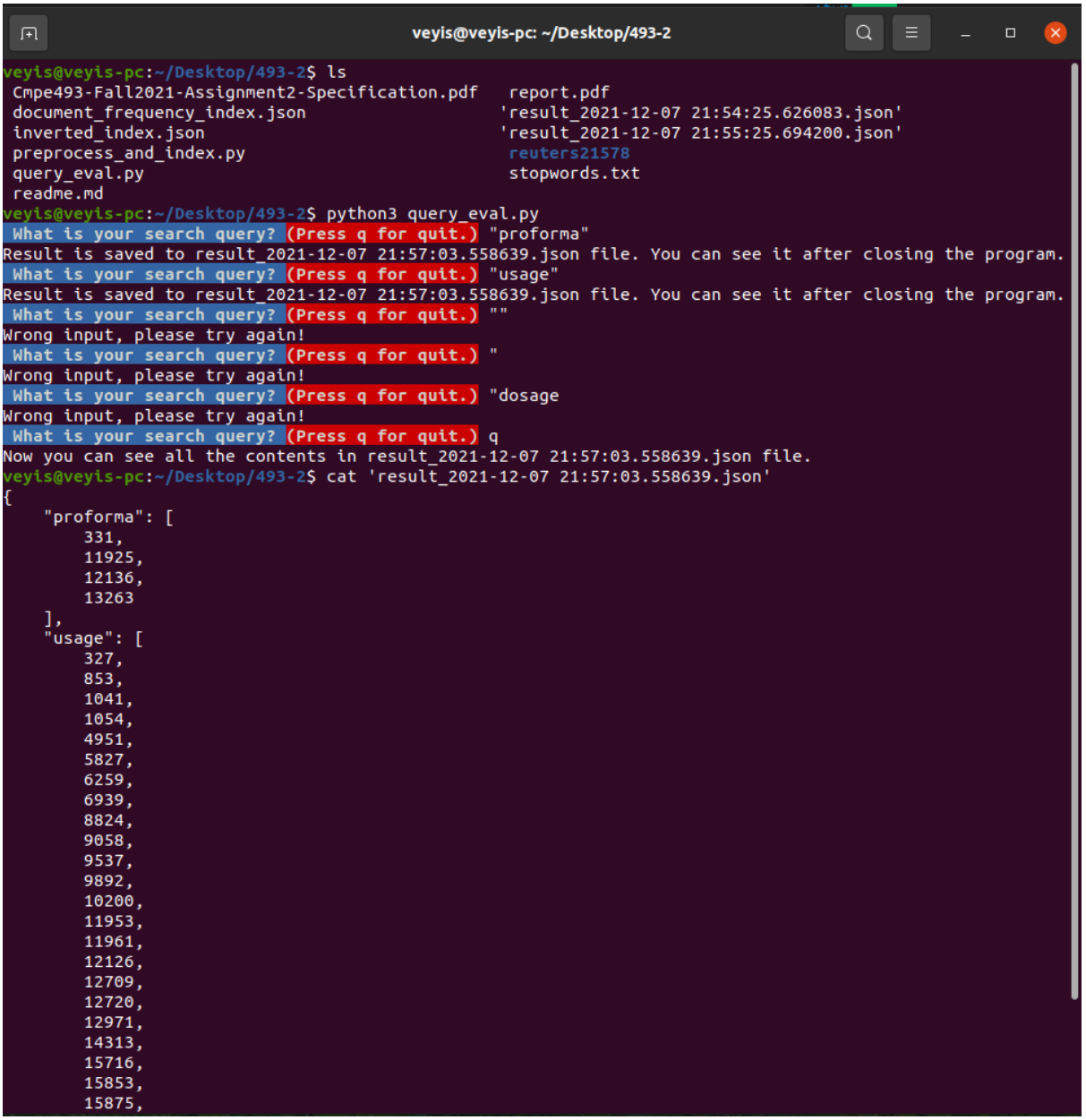
Note: At first, I printed out the exact locations of the phrases too. But, I realized that you only want document ids. So, I changed that to printing only document ids.

Adalet Veyis Turgut
2017400210
20/11/2021



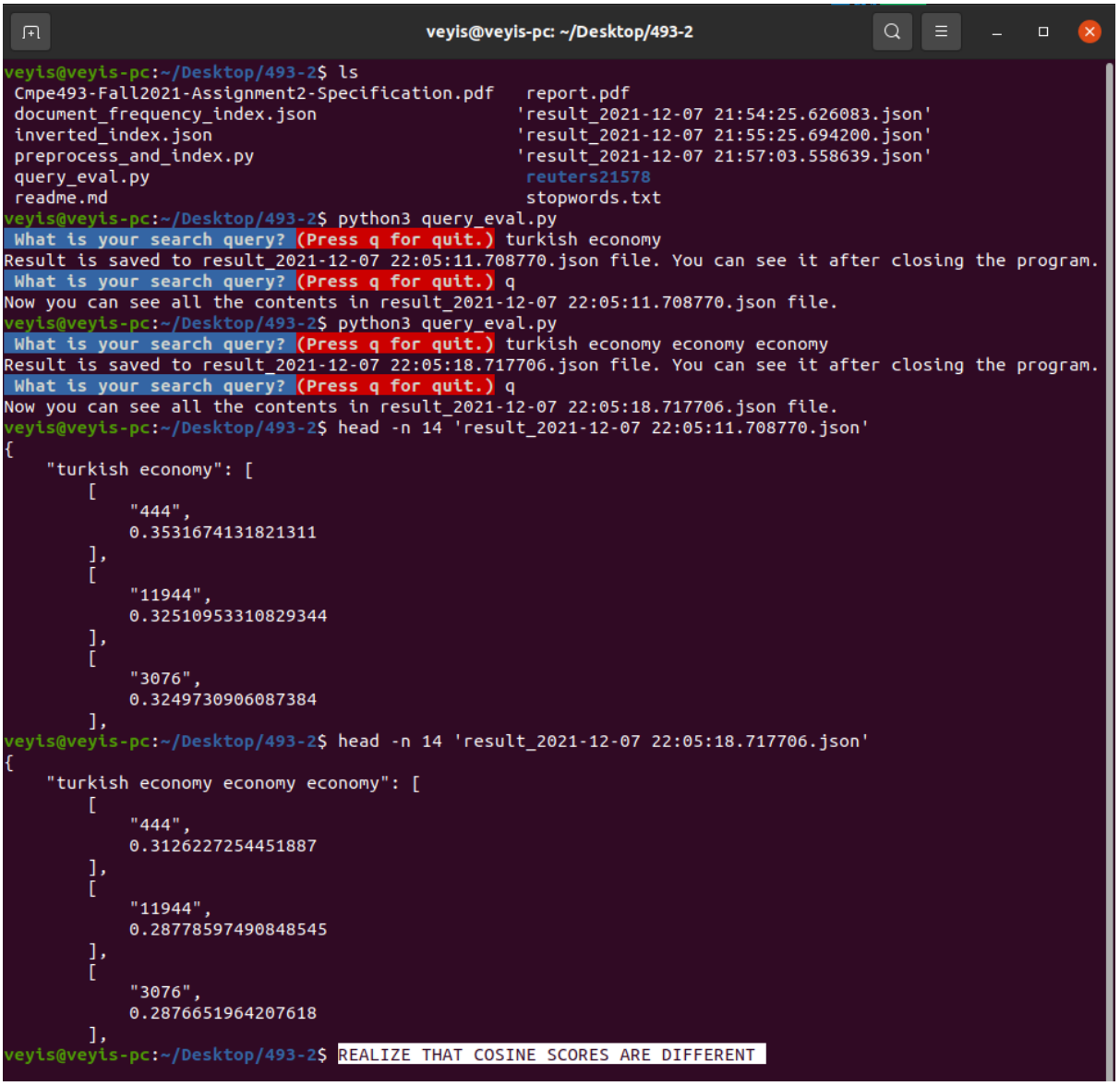


Adalet Veyis Turgut
2017400210
20/11/2021

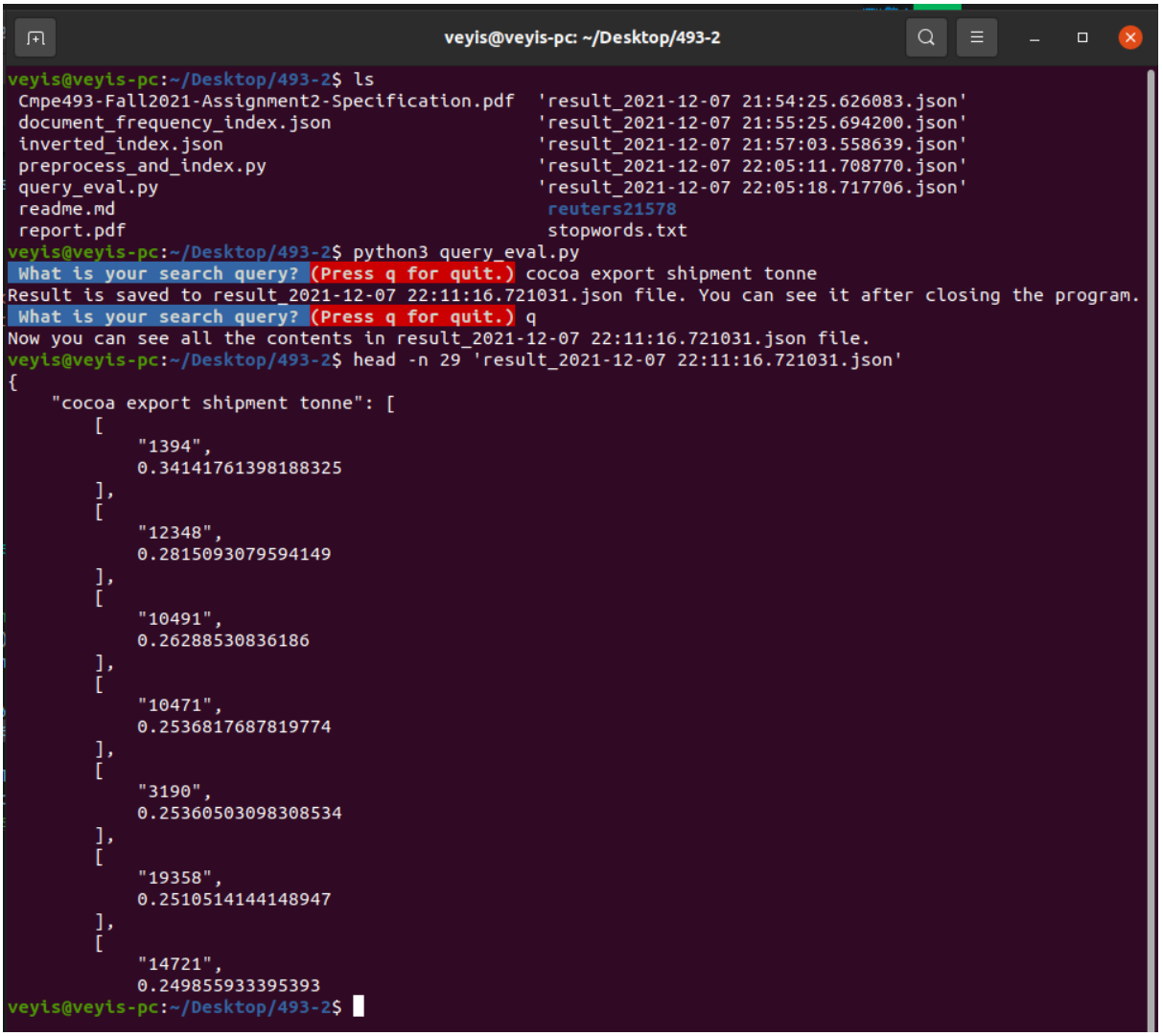


Adalet Veyis Turgut
2017400210
20/11/2021

4.2: free text



Adalet Veyis Turgut
2017400210
20/11/2021



Adalet Veyis Turgut
2017400210
20/11/2021

```
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$
```

Adalet Veyis Turgut
2017400210
20/11/2021