CMPE493 Introduction to Information Retrieval

Assignment 2 – A Simple Search System for Phrase and Free Text Queries

Ozan KILIÇ
2016400144

Report
-Data Preprocessing Steps
-Data Structures of Inverted Index
-Running Indexing Module
-Running Query Searches

1 Data Preprocessing Steps:

In the assignment, data preprocessing is handled separately at the tokenizer.py file. This file runs in the same folder with the sgm files that will be tokenized. At the beginning a list is created with the files ending as .sgm in the current folder. After parsing wanted files, there are two ways for preprocessing the data. As requested in the assignment description, the tokenizer shall also work separately. So, one of the modules is for running the file from terminal and the other one is for running the functionality from another class. These separate modules are do the same work at the end.

Before starting to iterate the sgm files, some tools for normalizing and parsing are defined. For parsing the body and title tags, the program looks for '<body>' and '<title>' words. One tool for this step is created with using regex library. re.compile(input) function returns a processed regex structure of the input and it is used while iterating the documents. The other tools are lists for punctuation and stop words.

At this stage iteration is started with iterateDoc module. This module is called for each sgm file separately. Strategy of the module is consist of two parts: search for a starting tag and save words until the ending tag. The searching operation is done with an iterator. When iterator finds starter tags, process jumps to the second part. In this part all words are normalized and saved until iterator catches an ending tag.

Words are saved in a list(tokenList) with document ids and index number in this step. Facing with ending tag returns the process to the first part for finding another starting tag. I want to mention an important point here. Although stop words are in the text, because we eliminated them previously in the project, in indexing step, they are again eliminated. Stop words has no effect on phrase search.

While iterating the documents, term frequencies of the words are also saved. I created a dictionary of dictionaries named 'docTfIdf' which takes document Ids as keys and the other dictionary as value. In the inner dictionary, terms are keys and their counts are values. Thanks to this data structure, at the end I created docTfIdf.json file in order to be used in cosine similarity calculation.

This process continues until iterator gives StopIteration exception. When exception occurs, iterateDoc() ends. In the module who calls iterateDoc, every tokenList list for all documents are stored in uniqueList list with eliminating the repeated elements in the list. The above process continues until all documents are parsed.

After all sgm files are iterated and tokens saved with their docIds, the list is first sorted accordingly token name and then token id. Now the list is ready for creating and inverted index dictionary(endDict). The sorted list is traversed and elements are added to the dictionary one by one.

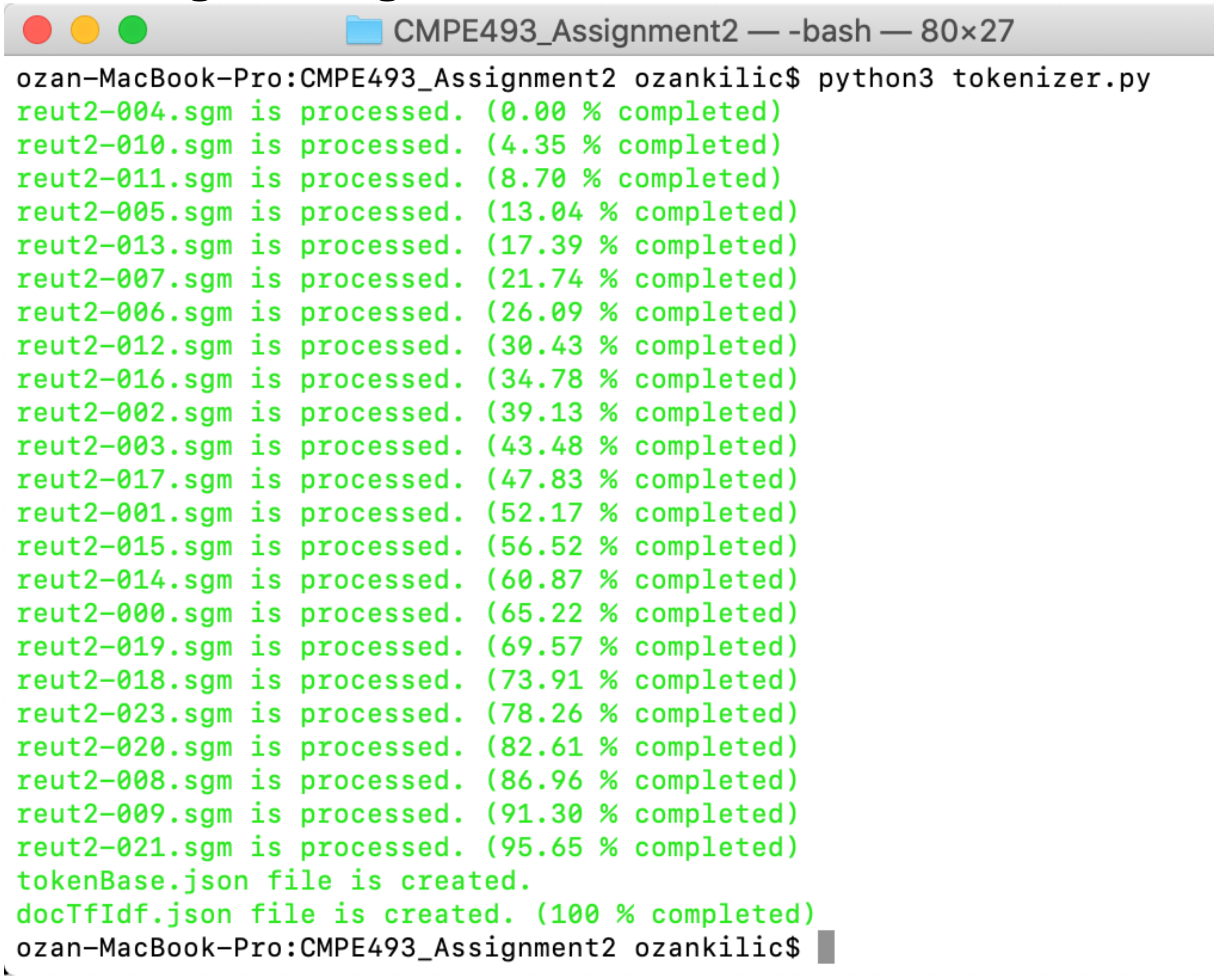
At the end, the inverted index dictionary is saved as a .json file named tokenBase.json. When searching query, searcher module reads that file and works on it.

2 Data Structures of Inverted Index:

Inverted index is composed of two parts: tokens and document ids. So, I saved them as a dictionary. Every token is unique and has docId values. Aim of this is making search faster with using python's dictionary's hashing functionality. In order to save document ids, list is used. So inverted index is a dictionary stores lists as values.

There is also another created file during tokenizing process. It is also a dictionary of dictionaries that stores documents' tf-idf scores named docTfIdf.json.

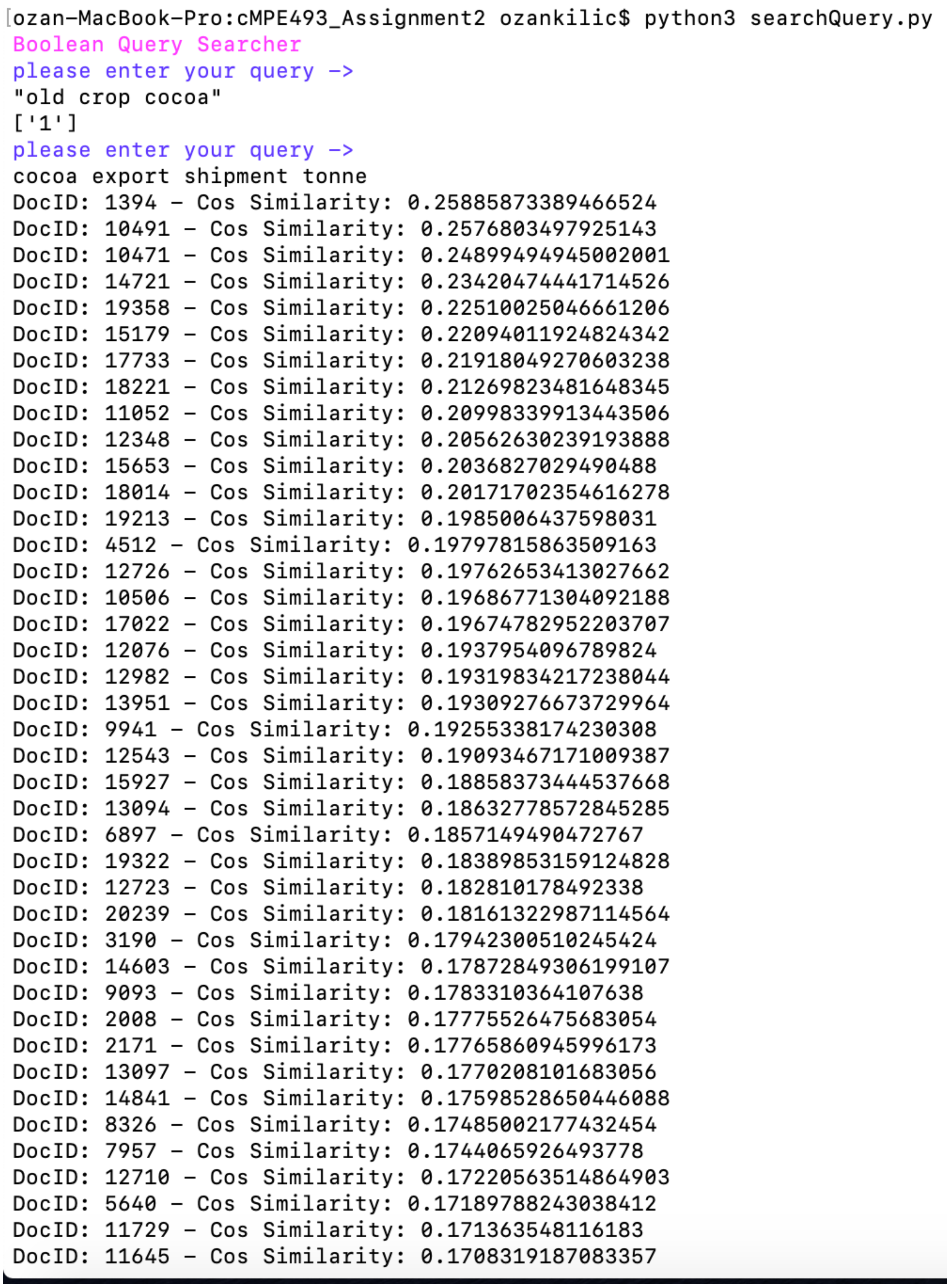
3 Running Indexing Module:



CMPE493_Assignment2 — -bash — 80×27

```
ozan-MacBook-Pro:CMPE493_Assignment2 ozankilic$ python3 tokenizer.py
reut2-004.sgm is processed. (0.00 % completed)
reut2-010.sgm is processed. (4.35 % completed)
reut2-011.sgm is processed. (8.70 % completed)
reut2-005.sgm is processed. (13.04 % completed)
reut2-013.sgm is processed. (17.39 % completed)
reut2-007.sgm is processed. (21.74 % completed)
reut2-006.sgm is processed. (26.09 % completed)
reut2-012.sgm is processed. (30.43 % completed)
reut2-016.sgm is processed. (34.78 % completed)
reut2-002.sgm is processed. (39.13 % completed)
reut2-003.sgm is processed. (43.48 % completed)
reut2-017.sgm is processed. (47.83 % completed)
reut2-001.sgm is processed. (52.17 % completed)
reut2-015.sgm is processed. (56.52 % completed)
reut2-014.sgm is processed. (60.87 % completed)
reut2-000.sgm is processed. (65.22 % completed)
reut2-019.sgm is processed. (69.57 % completed)
reut2-018.sgm is processed. (73.91 % completed)
reut2-023.sgm is processed. (78.26 % completed)
reut2-020.sgm is processed. (82.61 % completed)
reut2-008.sgm is processed. (86.96 % completed)
reut2-009.sgm is processed. (91.30 % completed)
reut2-021.sgm is processed. (95.65 % completed)
tokenBase.json file is created.
docTfIdf.json file is created. (100 % completed)
ozan-MacBook-Pro:CMPE493_Assignment2 ozankilic$
```

4 Running Query Searches:



```
ozan-MacBook-Pro:cMPE493_Assignment2 ozankilic$ python3 searchQuery.py
Boolean Query Searcher
please enter your query ->
"old crop cocoa"
['1']
please enter your query ->
cocoa export shipment tonne
DocID: 1394 - Cos Similarity: 0.25885873389466524
DocID: 10491 - Cos Similarity: 0.2576803497925143
DocID: 10471 - Cos Similarity: 0.24899494945002001
DocID: 14721 - Cos Similarity: 0.23420474441714526
DocID: 19358 - Cos Similarity: 0.22510025046661206
DocID: 15179 - Cos Similarity: 0.22094011924824342
DocID: 17733 - Cos Similarity: 0.21918049270603238
DocID: 18221 - Cos Similarity: 0.21269823481648345
DocID: 11052 - Cos Similarity: 0.20998339913443506
DocID: 12348 - Cos Similarity: 0.20562630239193888
DocID: 15653 - Cos Similarity: 0.2036827029490488
DocID: 18014 - Cos Similarity: 0.20171702354616278
DocID: 19213 - Cos Similarity: 0.1985006437598031
DocID: 4512 - Cos Similarity: 0.19797815863509163
DocID: 12726 - Cos Similarity: 0.19762653413027662
DocID: 10506 - Cos Similarity: 0.19686771304092188
DocID: 17022 - Cos Similarity: 0.19674782952203707
DocID: 12076 - Cos Similarity: 0.193795409678982
DocID: 12982 - Cos Similarity: 0.19319834217238044
DocID: 13951 - Cos Similarity: 0.19309276673729964
DocID: 9941 - Cos Similarity: 0.19255338174230308
DocID: 12543 - Cos Similarity: 0.19093467171009387
DocID: 15927 - Cos Similarity: 0.18858373444537668
DocID: 13094 - Cos Similarity: 0.18632778572845285
DocID: 6897 - Cos Similarity: 0.1857149490472767
DocID: 19322 - Cos Similarity: 0.18389853159124828
DocID: 12723 - Cos Similarity: 0.182810178492338
DocID: 20239 - Cos Similarity: 0.18161322987114564
DocID: 3190 - Cos Similarity: 0.17942300510245424
DocID: 14603 - Cos Similarity: 0.17872849306199107
DocID: 9093 - Cos Similarity: 0.1783310364107638
DocID: 2008 - Cos Similarity: 0.17775526475683054
DocID: 2171 - Cos Similarity: 0.17765860945996173
DocID: 13097 - Cos Similarity: 0.1770208101683056
DocID: 14841 - Cos Similarity: 0.17598528650446088
DocID: 8326 - Cos Similarity: 0.17485002177432454
DocID: 7957 - Cos Similarity: 0.1744065926493778
DocID: 12710 - Cos Similarity: 0.17220563514864903
DocID: 5640 - Cos Similarity: 0.17189788243038412
DocID: 11729 - Cos Similarity: 0.171363548116183
DocID: 11645 - Cos Similarity: 0.1708319187083357
```