CMPE492 ASSIGNMENT 1 REPORT

Data Preprocessing

Everything related to preprocessing and building the index is in the *preprocessor.py* file.

Get Documents as Row Strings

Preprocessing begins with *getDocuments()* method. In this method,

1) I will open a *.sgm* file and gather the whole text as a string one by one
2) After taking the content of one file as a string I will apply some regex operations:
    i. splitting with respect to the '*</REUTERS>*' tag: this operation will convert the whole string to a list of strings where each element of the list contains only one article(except the last element, since there is nothing after the last *</REUTERS>* tag), without the closing tag *</REUTERS>*. The following operations will be applied to every article string created with this step.
    ii. searching the pattern '*NEWID="[number]">*': this operation will get the id of the document(article) from the string.
    iii. searching the pattern '*<TEXT...<TITLE>[title_text]</TITLE>...':* to get the title of the document as a string.
    iv. searching the pattern '*<TEXT...<BODY>[body_text]</BODY>...*': to get the body of the document as a string
    v. searching the pattern '*<TEXT...[body_text]</TEXT>*': if title and body cannot be found in steps iii and iv, that means the article is in *UNPROC* format and it contains just text.
3) After step 2, we have a string that contains the whole text(either title+body or just text) as a string. I will insert that string into a global list to use it in other places.
4) Repeat the first 3 steps for every *.sgm* file in the *reuter21578* folder.

Normalize Tokens of Documents

After getting the document texts as raw strings, it is time to normalize those. The work is done in the *normalizeDocuments()* method:

1. split the documents into tokens by getting rid of whitespace characters.
2. normalizing the tokens of documents. this process is handled by a function in the *helper_function.py* file and contains several steps:
    i. checking whether the token is one of the clitics(we're, I've, etc.). clitics are defined in the *clitics.txt* file and that file is loaded with the helper function *getClitics()*. If the token is a clitic, then the token should be added to the resulting list after applying full case-folding.
    ii. if the token is not a clitic, the token should be shaved. the shaved version is either a word, words with hyphens included, or a token composed of numbers with special characters('/', ':', '.', ',') like 16/20, 19.02.2021, etc.
    iii. hyphens should be eliminated properly, if there are any. the methodology implied is as follows: if the first character of the hyphenated word is upper, like Hewett-Pickard or New York-San Fransisco, then they should be split and taken as different strings. if the first character is lower, then all hyphens should be deleted and the result will be one word('know-how' to 'knowhow').
    iv. if the token has a dot at the end, it is interpreted as the end of a sentence is found and the first character of the very following token should be lowered.
    v. the normalized tokens are added to a final list and that list with the number of the tokens gathered after the normalization step is returned.
3. gathering the normalized documents, which are lists of normalized tokens, in a list with their document ids
4. returning that list of normalized documents, number of tokens before normalization, number of tokens after normalization, and number of terms before normalization

Create the Dictionary

Since the tokens in documents are normalized, the dictionary of the corpus can be created. *getDictionary()* method deals with that problem. It simply creates an empty set and tries to add every token to that set. Since the set data structure does not allow duplicate entries, we have our dictionary. After it is sorted, data preprocessing is finished.

Preprocess Metrics

As it is written in the *preprocessing_metrics.txt* file,

- The number of tokens before normalization is **130957**
- The number of tokens after normalization: **2760537**
- The number of terms before normalization: **130957**
- The number of terms after normalization: **85248**

The top 100 most frequent terms after all normalization steps -which are case-folding, removal of punctuations, and getting rid of hyphens- are given in the ***top_100_frequent_terms.txt***.

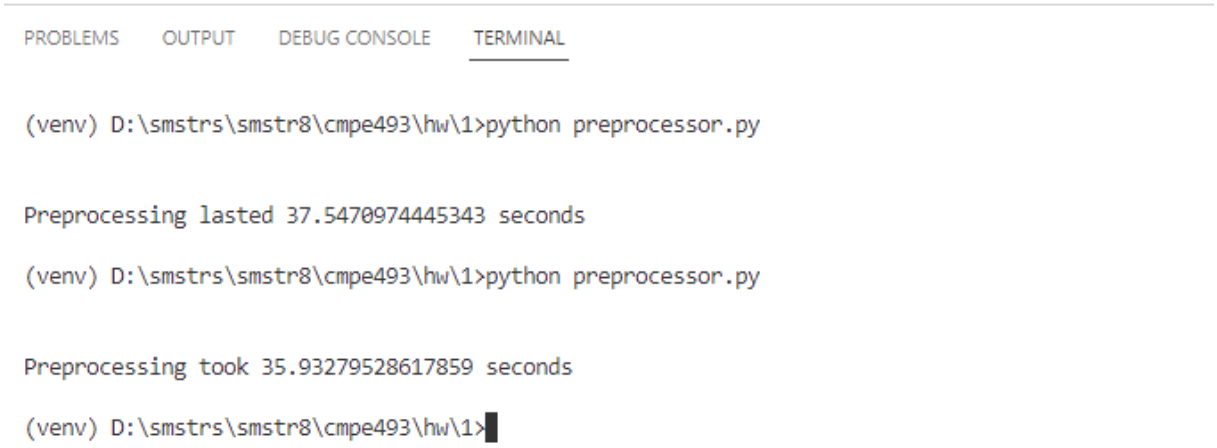
The whole process of handling the preprocessing and building the index takes around **35 seconds** on my computer.

Indexing

I have kept the dictionary and the postings lists separately, as lists. The dictionary is just a list of strings, where every string is a term. Postings lists are arranged so that the position index of a term in the dictionary is the same as the position index of its corresponding postings lists. Postings lists are stored as tuples in a list: the first element of the tuple is the term frequency and the second element is a dictionary(map) that maps the document ids(keys) to the list of position indexes(values) where the term occurs in that document.

Screenshots

Preprocessing



Query Processing