Name: Ghaidaa Ali Alyoubi

ID: 3950585

Section: ID9G

Lab#7(Positional Indexing)

Exercise5. For the whole collection given by the instructor, build the positional index and find frequency for each term appears in collection!

The code:

```
import numpy as np
import os
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import TweetTokenizer
from natsort import natsorted
import string

def read_file(filename):
    with open(filename, 'r', encoding ="ascii", errors ="surrogateescape") as f:
        stuff = f.read()

    f.close()

    # Remove header and footer.
    stuff = remove_header_footer(stuff)

    return stuff

def remove_header_footer(final_string):
    new_final_string = ""
    tokens = final_string.split('\n\n')

    # Remove tokens[0] and tokens[-1]
    for token in tokens[1:-1]:
        new_final_string += token+" "
```

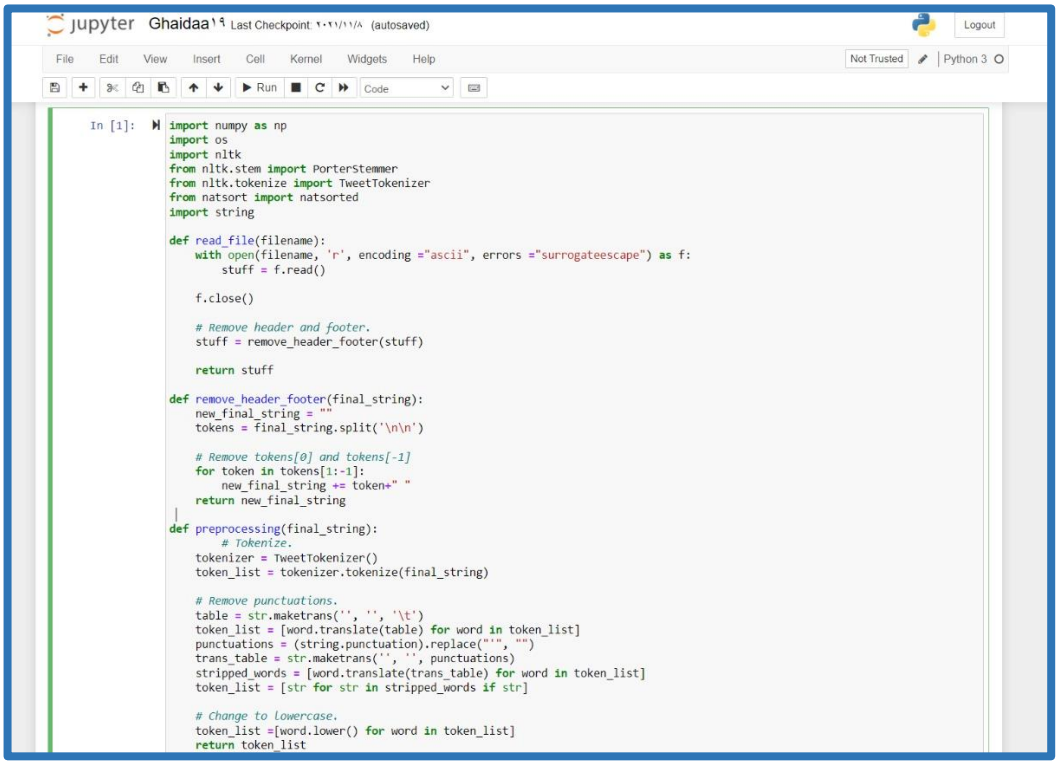
```
    return new_final_string

def preprocessing(final_string):
        # Tokenize.
    tokenizer = TweetTokenizer()
    token_list = tokenizer.tokenize(final_string)

    # Remove punctuations.
    table = str.maketrans('', '', '\t')
    token_list = [word.translate(table) for word in token_list]
    punctuations = (string.punctuation).replace("'", "")
    trans_table = str.maketrans('', '', punctuations)
    stripped_words = [word.translate(trans_table) for word in token_list]
    token_list = [str for str in stripped_words if str]

    # Change to lowercase.
    token_list =[word.lower() for word in token_list]
    return token_list

# In this example, we create the positional index for only 1 folder.
folder_names = ["comp.graphics"]

# Initialize the stemmer.
stemmer = PorterStemmer()

# Initialize the file no.
fileno = 0

# Initialize the dictionary.
pos_index = {}
```

```
# Initialize the file mapping (fileno -> file name).
file_map = {}

for folder_name in folder_names:

    # Open files.
    file_names = natsorted(os.listdir("C:/Users/luluu/newsgroups/" + folder_name))

    # For every file.
    for file_name in file_names:

        # Read file contents.
        stuff = read_file("C:/Users/luluu/newsgroups/" + folder_name + "/" + file_name)

        # This is the list of words in order of the text.
        # We need to preserve the order because we require positions.
        # 'preprocessing' function does some basic punctuation removal,
        # stopword removal etc.
        final_token_list = preprocessing(stuff)

        # For position and term in the tokens.
        for pos, term in enumerate(final_token_list):

            # First stem the term.
            term = stemmer.stem(term)

            # If term already exists in the positional index dictionary.
            if term in pos_index:

                # Increment total freq by 1.
                pos_index[term][0] = pos_index[term][0] + 1
```

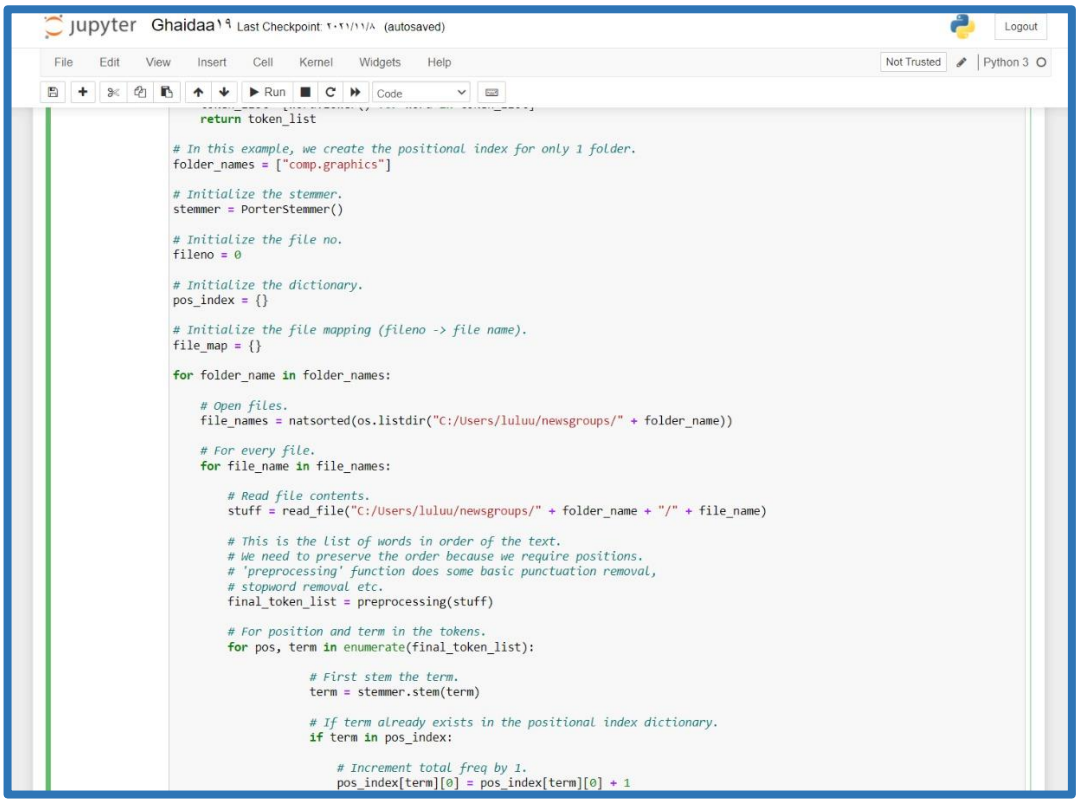
```
                # Check if the term has existed in that DocID before.
                if fileno in pos_index[term][1]:
                    pos_index[term][1][fileno].append(pos)

                else:
                    pos_index[term][1][fileno] = [pos]

            # If term does not exist in the positional index dictionary
            # (first encounter).
            else:

                # Initialize the list.
                pos_index[term] = []
                # The total frequency is 1.
                pos_index[term].append(1)
                # The postings list is initially empty.
                pos_index[term].append({})
                # Add doc ID to postings list.
                pos_index[term][1][fileno] = [pos]

        # Map the file no. to the file name.
        file_map[fileno] = "newsgroups/" + folder_name + "/" + file_name

        # Increment the file no. counter for document ID mapping
        fileno += 1

sample_pos_idx = pos_index["andrew"]
print("Positional Index")
print(sample_pos_idx)
```

```
file_list = sample_pos_idx[1]

print("Filename, [Positions]")

for fileno, positions in file_list.items():

    print(file_map[fileno], positions)
```

The Screenshots:



```
In [1]:
import numpy as np
import os
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import TweetTokenizer
from natsort import natsorted
import string

def read_file(filename):
    with open(filename, 'r', encoding ="ascii", errors ="surrogateescape") as f:
        stuff = f.read()

    f.close()

    # Remove header and footer.
    stuff = remove_header_footer(stuff)

    return stuff

def remove_header_footer(final_string):
    new_final_string = ""
    tokens = final_string.split('\n\n')

    # Remove tokens[0] and tokens[-1]
    for token in tokens[1:-1]:
        new_final_string += token+" "
    return new_final_string

def preprocessing(final_string):
        # Tokenize.
    tokenizer = TweetTokenizer()
    token_list = tokenizer.tokenize(final_string)

    # Remove punctuations.
    table = str.maketrans('', '', '\t')
    token_list = [word.translate(table) for word in token_list]
    punctuations = (string.punctuation).replace("'", "")
    trans_table = str.maketrans('', '', punctuations)
    stripped_words = [word.translate(trans_table) for word in token_list]
    token_list = [str for str in stripped_words if str]

    # Change to lowercase.
    token_list =[word.lower() for word in token_list]
    return token_list
```



```
    return token_list

# In this example, we create the positional index for only 1 folder.
folder_names = ["comp.graphics"]

# Initialize the stemmer.
stemmer = PorterStemmer()

# Initialize the file no.
fileno = 0

# Initialize the dictionary.
pos_index = {}

# Initialize the file mapping (fileno -> file name).
file_map = {}

for folder_name in folder_names:

    # Open files.
    file_names = natsorted(os.listdir("C:/Users/luluu/newsgroups/" + folder_name))

    # For every file.
    for file_name in file_names:

        # Read file contents.
        stuff = read_file("C:/Users/luluu/newsgroups/" + folder_name + "/" + file_name)

        # This is the list of words in order of the text.
        # We need to preserve the order because we require positions.
        # 'preprocessing' function does some basic punctuation removal,
        # stopword removal etc.
        final_token_list = preprocessing(stuff)

        # For position and term in the tokens.
        for pos, term in enumerate(final_token_list):

                # First stem the term.
                term = stemmer.stem(term)

                # If term already exists in the positional index dictionary.
                if term in pos_index:

                    # Increment total freq by 1.
                    pos_index[term][0] = pos_index[term][0] + 1
```

jupyter Ghaidaa١٩ Last Checkpoint: ٢٠٢١/١١/٨ (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```python
                    pos_index[term][1][fileno].append(pos)

                else:
                    pos_index[term][1][fileno] = [pos]

            # If term does not exist in the positional index dictionary
            # (first encounter).
            else:

                # Initialize the list.
                pos_index[term] = []
                # The total frequency is 1.
                pos_index[term].append(1)
                # The postings list is initially empty.
                pos_index[term].append({})
                # Add doc ID to postings list.
                pos_index[term][1][fileno] = [pos]

        # Map the file no. to the file name.
        file_map[fileno] = "newsgroups/" + folder_name + "/" + file_name

        # Increment the file no. counter for document ID mapping
        fileno += 1

sample_pos_idx = pos_index["andrew"]
print("Positional Index")
print(sample_pos_idx)

file_list = sample_pos_idx[1]
print("Filename, [Positions]")
for fileno, positions in file_list.items():
    print(file_map[fileno], positions)
```

```
Positional Index
[10, {215: [2081], 539: [66], 591: [879], 616: [462, 473], 680: [135], 691: [2081], 714: [4], 809: [333], 979: [0]}]
Filename, [Positions]
newsgroups/comp.graphics/38376 [2081]
newsgroups/comp.graphics/38701 [66]
newsgroups/comp.graphics/38753 [879]
newsgroups/comp.graphics/38778 [462, 473]
newsgroups/comp.graphics/38842 [135]
newsgroups/comp.graphics/38853 [2081]
newsgroups/comp.graphics/38876 [4]
newsgroups/comp.graphics/38971 [333]
newsgroups/comp.graphics/39663 [0]
```