cmpe 493 Assignment #2

Kayacan Vesek 2016400114

December 11, 2020

(i)Describe the steps you have performed for data preprocessing. (ii) Describe the data structures that you used for representing the inverted index. (iii) Describe the trie data structure that you used in your code and provide your well- commented trie code in the report.

1 Preprocessing Steps - Obtaining Words

I read all the stopwords, and read all the text bodies and titles with using regex
I did following steps on title and body:
erase all punctuations and replaced with ' '
and split the sentences to words using string.split(). than I add all words to trie.

```python
stopWords = [i.rstrip().casefold() for i in open('stopwords.txt', 'r', encoding='latin-1').readlines()]
for title,body in ALL_TITLE_AND_BODIES:

    title = title.translate(str.maketrans(string.punctuation, len(string.punctuation)*' '))
    body = body.translate(str.maketrans(string.punctuation, len(string.punctuation)*' '))

    for word in set(((title + ' ' + body).casefold().split())):
        if word not in stopWords:
            #ADD THIS WORDS TO TRIE
            #ADD this wort to Inverted Index
```

2 Inverted Index

JSON is a data format (a string), Python dictionary is a data structure (in-memory object). I used dictionary data structure in python, it is easy to convert to json. A Dictionary in Python is the unordered and changeable collection of data values that holds key-value pairs. Each key-value pair in the dictionary maps the key to its associated value making it more optimized.

So there is a defined array object for each word.
index['word'] is defined as array so if index['word'] = [1,5,7]. This means 'word' can be found in indexes [1,5,7].

```python
index[word].append(index) #this one line is enough to add the word.
```

3 Trie

This Trie implementation is the shortest and easiest(to who is expert) to implement, it is not so easy to understand who are not familiar with Trie.

Tree is simply is array of maps(dicts in python).
Each object in the array is a dictionary(one dictionary for each node).
So in the end if there are N nodes, array length will be N
In each dictionary edges will kept like 'A':11 , 'C':3 So this means there is an edge with char 'A' to node 11(11. index of this array)

```
tree = []    # this one keeps the nodes and paths from this node
tree.append(dict())
tree_ans = []     # this one keeps the answers for this node, for example root has all the words as an answer
tree_ans.append([])
node = 0 ## number of nodes in trie
def trieAdd(word):
    now = 0 # Current Node, 0 is the root
    tree_ans[now].append(index)
    for j in word: # itterate over the word, j is a char
        if j not in tree[now].keys(): # check if edge exists
            node += 1
            tree_ans.append([])
            tree.append(dict())
            tree[now][j] = node

        now=tree[now][j] # Go to edge with using edge j
        tree_ans[now].append(index) ## add the index of word to the trie node.
```