AIR Assignment Dataset-3

Sreenath Saikumar PES2UG19CS406

Srivatsan Narendiran PES2UG19CS408

Sreekanth Maneesh PES2UG19CS405

Link to the Code used in the assignment:
https://github.com/SreenathSaikumar/AIR_assignment

3 Datasets were chosen to be used for the assignment:

- Legal Citation Text Classification
- Shopee Text Reviews
- Emotion Detection from Text

The first 2 datasets were used to demonstrate and benchmark phrase querying with proximity while the 3rd dataset was used to perform simple Boolean queries with AND, OR and NOT.

Benchmarks:

Both Positional index creation or Inverted index creation in the case of the 3rd dataset as well as query retrieval times were benchmarked to infer an approximate performance comparison.

benchmark.txt

You, 7 minutes ago | 1 author (You)

```
1  =====Tweet Emotions Dataset=====      You, 7 minutes ago
2  Time taken to build inverted index: 0.09222626686096191
3  Time taken to search: 0.0012993812561035156
4  =====Legal Text Classification Dataset=====
5  Time taken to build inverted index: 5.549209117889404
6  Time taken to search for phrase: 0.03466200828552246
7  =====Shopee Review Dataset=====
8  Time taken to build inverted index: 20.161844968795776
9  Time taken to search for phrase: 0.39845800399780273
10
```

This file was generated while running the index construction and queries on all the datasets

Code for Dataset 3 (Text Emotions):

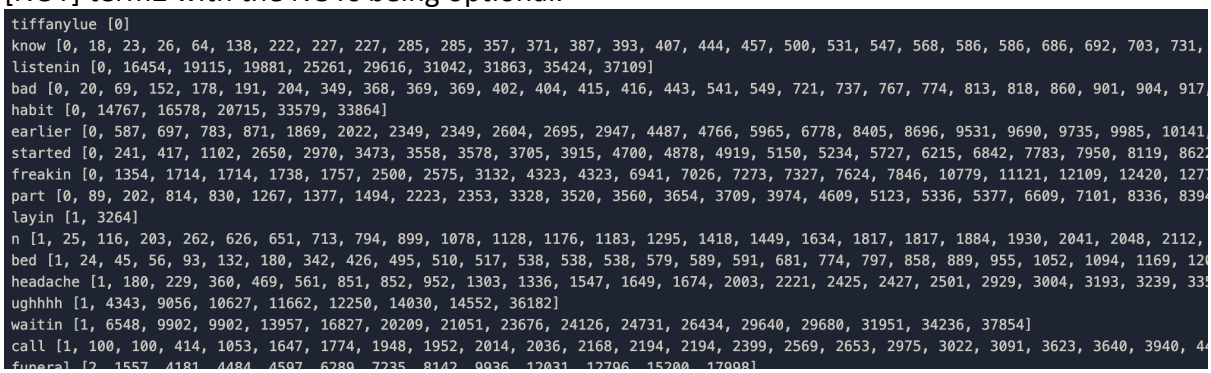
Boolean querying has been performed

```
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
import nltk

from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import pandas as pd
import numpy as np
import time

def intersect(p1,p2):
    res=[]
    i,j=0,0
    while(i<len(p1) and j<len(p2)):
        if(p1[i]==p2[j]):
            res.append(p1[i])
            i+=1
            j+=1
        elif p1[i]>p2[j]:
            j+=1
        else:
            i+=1
    return res

def postunion(p1,p2):
    res=[]
    res=sorted(list(set().union(p1,p2)))
    return res

def notquery(dfsize,p1):
    res=[]
    for i in range(dfsize):
        if i not in p1:
            res.append(i)
    return res

df=pd.read_csv('tweet_emotions.csv')
stop_words=set(stopwords.words('english'))
df['texttoken']=df['content'].apply(word_tokenize)
df['texttoken']=df['texttoken'].apply(lambda words:[word.lower() for word in words
if word.isalpha()])
df['stop_remd']=df['texttoken'].apply(lambda x:[item for item in x if item not in
stop_words])
st_time=time.time()
post_list={}
for postext,text in enumerate(df['stop_remd']):
    for pos,term in enumerate(text):
```

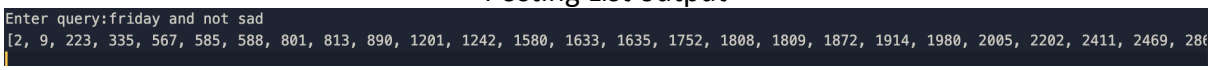
```
        if term not in post_list.keys():
            post_list[term]=[]
        post_list[term].append(postext)
end_time=time.time()-st_time
f=open("benchmark.txt",'a')
f.write("=====Tweet Emotions Dataset=====\n")
f.write("Time taken to build inverted index: "+str(end_time)+"\n")

def querysearch(inp,post_list):
    if len(inp)==3:
        if inp[1].lower() == 'and':
            print(intersect(post_list[inp[0]],post_list[inp[2]]))
        elif inp[1].lower() == 'or':
            print(postunion(post_list[inp[0]],post_list[inp[2]]))
        else:
            print("Invalid query")
    elif len(inp)==4:
        if inp[0].lower() == 'not' and inp[2].lower() == 'and':

print(intersect(notquery(df.shape[0],post_list[inp[1]]),post_list[inp[3]]))
        elif inp[0].lower() == 'not' and inp[2].lower() == 'or':

print(postunion(notquery(df.shape[0],post_list[inp[1]]),post_list[inp[3]]))
        elif inp[2].lower() == 'not' and inp[1].lower() == 'and':

print(intersect(notquery(df.shape[0],post_list[inp[3]]),post_list[inp[0]]))
        elif inp[2].lower() == 'not' and inp[1].lower() == 'or':

print(postunion(notquery(df.shape[0],post_list[inp[3]]),post_list[inp[0]]))
        else:
            print("Invalid query")
    elif len(inp)==5:
        if inp[2].lower()=='and':
            p1=notquery(df.shape[0],post_list[inp[1]])
            p2=notquery(df.shape[0],post_list[inp[4]])
            print(intersect(p1,p2))
        elif inp[2].lower()=='or':

print(postunion(notquery(df.shape[0],post_list[inp[1]]),notquery(df.shape[0],post_l
ist[inp[4]])))
        else:
            print("Invalid query")
    else:
        print("Invalid query")

inp=input("Enter query:").split()
st_time=time.time()
querysearch(inp,post_list)
end_time=time.time()-st_time
f.write("Time taken to search: "+str(end_time)+"\n")
f.close()
```

Outputs:

The input to the program is a simple Boolean query of the form [NOT] term1 [AND|OR] [NOT] term2 with the NOTs being optional.



Posting List output



Query output

Once the user enters the query, the output is a list of all the documents that satisfy the Boolean query.