AIR Assignment Dataset-2

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 2 (Shopee Reviews):

Phrase Querying with proximity 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 time


df=pd.read_csv('shopee_reviews.csv',usecols=["text"])
stop_words=set(stopwords.words('english'))
df=df.dropna()
df['texttoken']=df['text'].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])

ps=PorterStemmer()
df['stemmed']=df['stop_remd'].apply(lambda x:[ps.stem(item) for item in x])


idx_dict={}
st_time=time.time()
for postext,text in enumerate(df['stemmed']):
    for pos,term in enumerate(text):
        if term not in idx_dict.keys():
            idx_dict[term]=[0,{}]
        idx_dict[term][0]+=1
        if postext not in idx_dict[term][1].keys():
            idx_dict[term][1][postext]=[]
        idx_dict[term][1][postext].append(pos)
end_time=time.time()-st_time
f=open("benchmark.txt",'a')
f.write("=====Shopee Review Dataset=====\n")
f.write("Time taken to build inverted index: "+str(end_time)+"\n")
print("Term [Frequency,Entry number:[Positions in that entry]]")
for i in list(idx_dict)[:10]:
    print(i,idx_dict[i])

def search(t1,t2,prox,idx_dict):
    res=[]
    if t1 not in idx_dict.keys() or t2 not in idx_dict.keys():
        print("Query does not exist")
    else:
        idx1=idx_dict[t1][1]
        idx2=idx_dict[t2][1]
```

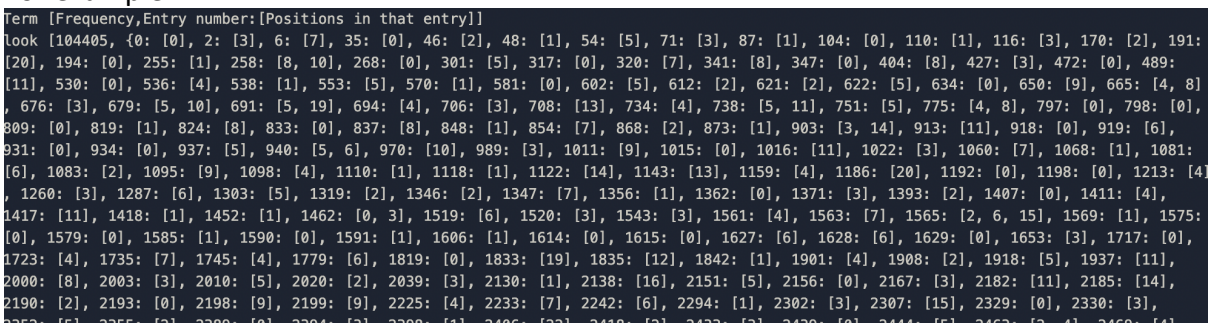
```
        interset=set(idx1.keys()).intersection(idx2.keys())
        for i in interset:
            l1=idx1[i]
            l2=idx2[i]
            for j in l1:
                if (any(x in l2 for x in range(j-prox,j+prox+1))):
                    if i not in res:
                        res.append(i)
    if len(res)==0:
        print('Query does not exist')
    return res

inp=input("Enter query with query proximity separated by spaces:").split()
prox=1
if(len(inp)==3):
  prox=int(inp[1][1:])
  inp.pop(1)
inp=[ps.stem(i) for i in inp]
st_time=time.time()
res=search(inp[0],inp[1],prox,idx_dict)
end_time=time.time()-st_time
f.write("Time taken to search for phrase: "+str(end_time)+"\n")
f.close()
print(res)
if len(res)!=0:
    for i in res:
        print(i,"\t",df['text'].iloc[i])
```

Outputs:

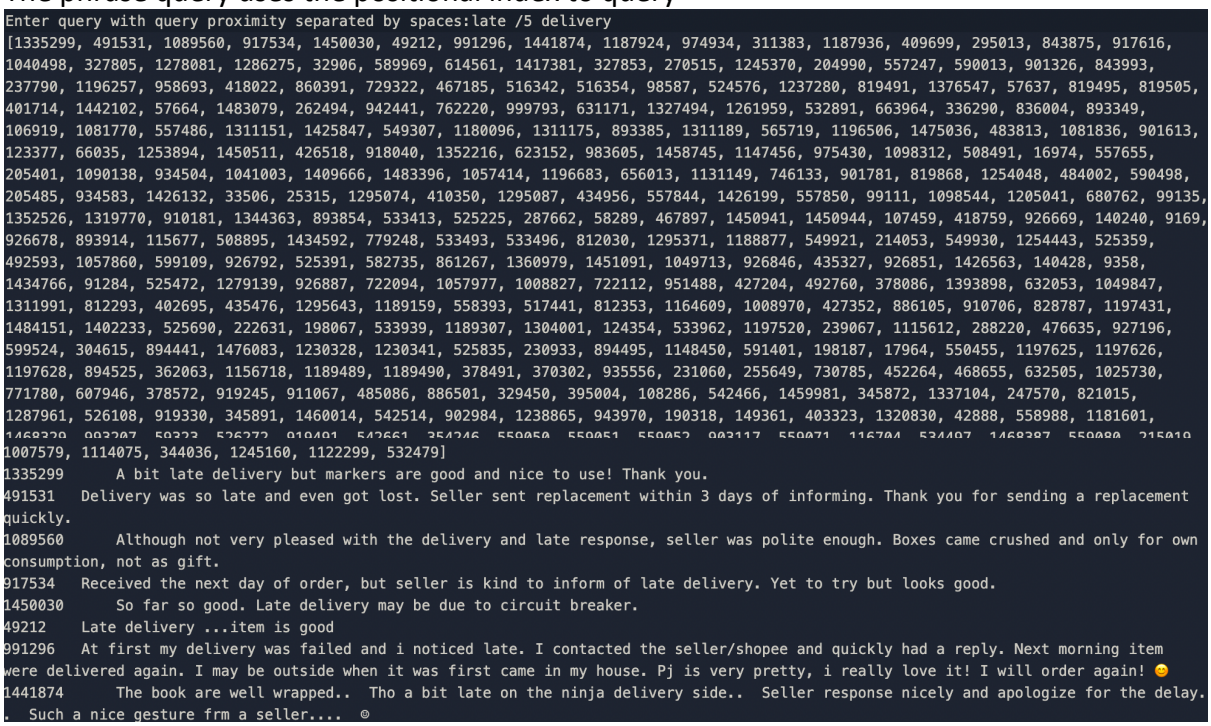
Positional index outputs are of the form ‘Term [Total Frequency, Entry number:[Positions in that entry]]’ in the output

For example:



Snapshot of the positional index

The phrase query uses the positional index to query



Snapshot of the output

The output of the phrase query is a list of all the matching documents followed by the contents of each document.