AIR Assignment Dataset-3

Sapna Singh PES2UG19CS366

Siddharth Chattar PES2UG19CS388

Sneha Sujit Saha PES2UG19CS393

Link to the Code used in the assignment:
https://github.com/sapnasingh2041/AIWIR_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

```python
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):
```

```python
        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.

```
tiffanylue [0]
know [0, 18, 23, 26, 64, 138, 222, 227, 227, 285, 285, 357, 371, 387, 393, 407, 444, 457, 500, 531, 547, 568, 586, 586, 686, 692, 703, 731,
listenin [0, 16454, 19115, 19881, 25261, 29616, 31042, 31863, 35424, 37109]
bad [0, 20, 69, 152, 178, 191, 204, 349, 368, 369, 369, 402, 404, 415, 416, 443, 541, 549, 721, 737, 767, 774, 813, 818, 860, 901, 904, 917
habit [0, 14767, 16578, 20715, 33579, 33864]
earlier [0, 587, 697, 783, 871, 1869, 2022, 2349, 2349, 2604, 2695, 2947, 4487, 4766, 5965, 6778, 8405, 8696, 9531, 9690, 9735, 9985, 10141
started [0, 241, 417, 1102, 2650, 2970, 3473, 3558, 3578, 3705, 3915, 4700, 4878, 4919, 5150, 5234, 5727, 6215, 6842, 7783, 7950, 8119, 8622
freakin [0, 1354, 1714, 1714, 1738, 1757, 2500, 2575, 3132, 4323, 4323, 6941, 7026, 7273, 7327, 7624, 7846, 10779, 11121, 12109, 12420, 1277
part [0, 89, 202, 814, 830, 1267, 1377, 1494, 2223, 2353, 3328, 3520, 3560, 3654, 3709, 3974, 4609, 5123, 5336, 5377, 6609, 7101, 8336, 8394
layin [1, 3264]
n [1, 25, 116, 203, 262, 626, 651, 713, 794, 899, 1078, 1128, 1176, 1183, 1295, 1418, 1449, 1634, 1817, 1817, 1884, 1930, 2041, 2048, 2112,
bed [1, 24, 45, 56, 93, 132, 180, 342, 426, 495, 510, 517, 538, 538, 538, 579, 589, 591, 681, 774, 797, 858, 889, 955, 1052, 1094, 1169, 120
headache [1, 180, 229, 360, 469, 561, 851, 852, 952, 1303, 1336, 1547, 1649, 1674, 2003, 2221, 2425, 2427, 2501, 2929, 3004, 3193, 3239, 335
ughhhh [1, 4343, 9056, 10627, 11662, 12250, 14030, 14552, 36182]
waitin [1, 6548, 9902, 9902, 13957, 16827, 20209, 21051, 23676, 24126, 24731, 26434, 29640, 29680, 31951, 34236, 37854]
call [1, 100, 100, 414, 1053, 1647, 1774, 1948, 1952, 2014, 2036, 2168, 2194, 2194, 2399, 2569, 2653, 2975, 3022, 3091, 3623, 3640, 3940, 44
funeral [2, 1557, 4181, 4484, 4597, 6289, 7235, 8142, 9936, 12031, 12796, 15200, 17998]
```

Posting List output

```
Enter query:friday and not sad
[2, 9, 223, 335, 567, 585, 588, 801, 813, 890, 1201, 1242, 1580, 1633, 1635, 1752, 1808, 1809, 1872, 1914, 1980, 2005, 2202, 2411, 2469, 286
```

Query output

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