AIWIR ASSIGNMENT-2

SEC: 6E

Submitted by:

Name	SRN
Rishab Kshatri	PES2UG19CS327
Rahul S Bhat	PES2UG19CS315
Rahul Radhakrishna	PES2UG19CS314

Bi word phrase query retrieval using inverted index and positional index

CODE:

In [1]:
```
import pandas
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
import string
import time
stemmer = PorterStemmer()
```

In [2]:
```
data=pandas.read_csv("C:/Users/Rahul Bhat/Desktop/file_name.csv")
data.drop(['Score','Id','Subreddit','URL','Num of Comments','Text','Date Created'], axis=1, inplace=True)
data.head()
```

Out[2]:

	Title	Political Lean
0	No matter who someone is, how they look like, ...	Liberal
1	Biden speech draws 38.2 million U.S. TV viewers	Liberal
2	State of the union	Liberal
3	We Should Just Give Poor People Money	Liberal
4	Do it for the Dew	Liberal

In [3]:
```
sentences = []
words = []
filtered = []
```

In [4]:
```
for i in data.Title:
        i = i.translate(str.maketrans('', '', string.punctuation))
        a = sent_tokenize(i)
        sentences.append(a)
        b = word_tokenize(i)
        words.append(b)
        c = [w.lower() for w in b]
        filtered.append(c)
data.head()
```

Out[4]:

	Title	Political Lean
0	No matter who someone is, how they look like, ...	Liberal
1	Biden speech draws 38.2 million U.S. TV viewers	Liberal
2	State of the union	Liberal
3	We Should Just Give Poor People Money	Liberal
4	Do it for the Dew	Liberal

In [5]:
```
data['sentence_tokenized'] = sentences
data['word_tokenized'] = words
data['filtered'] = filtered
data.head()
```

Out[5]:

	Title	Political Lean	sentence_tokenized	word_tokenized	filtered
0	No matter who someone is, how they look like, ...	Liberal	[No matter who someone is how they look like w...	[No, matter, who, someone, is, how, they, look...	[no, matter, who, someone, is, how, they, look...
1	Biden speech draws 38.2 million U.S. TV viewers	Liberal	[Biden speech draws 382 million US TV viewers]	[Biden, speech, draws, 382, million, US, TV, v...	[biden, speech, draws, 382, million, us, tv, v...

In [6]:
```
data['stemmed_words']=data['filtered'].apply(lambda f: [stemmer.stem(word) for word in f])
data.head()
```

Out[6]:

	Title	Political Lean	sentence_tokenized	word_tokenized	filtered	stemmed_words
0	No matter who someone is, how they look like, ...	Liberal	[No matter who someone is how they look like w...	[No, matter, who, someone, is, how, they, look...	[no, matter, who, someone, is, how, they, look...	[no, matter, who, someon, is, how, they, look,...
1	Biden speech draws 38.2 million U.S. TV viewers	Liberal	[Biden speech draws 382 million US TV viewers]	[Biden, speech, draws, 382, million, US, TV, v...	[biden, speech, draws, 382, million, us, tv, v...	[biden, speech, draw, 382, million, us, tv, vi...
2	State of the union	Liberal	[State of the union]	[State, of, the, union]	[state, of, the, union]	[state, of, the, union]
3	We Should Just Give Poor People Money	Liberal	[We Should Just Give Poor People Money]	[We, Should, Just, Give, Poor, People, Money]	[we, should, just, give, poor, people, money]	[we, should, just, give, poor, peopl, money]
4	Do it for the Dew	Liberal	[Do it for the Dew]	[Do, it, for, the, Dew]	[do, it, for, the, dew]	[do, it, for, the, dew]

In [8]:
```
pos_index = {}
for docid,tList in enumerate(data['stemmed_words']):
    for position,w in enumerate(tList):
        if w in pos_index.keys():
            pos_index[w][0]+=1
            if docid in pos_index[w][1]:
                pos_index[w][1][docid].append(position)
            else:
                pos_index[w][1][docid]=[position]
        else:
            temp={docid : [position]}
            pos_index[w]=[1,temp]
```

In [9]:
```
for i in pos_index:
    print(i,":",pos_index[i],end='\n\n')
```

```
no : [250, {0: [0], 31: [16], 46: [0], 93: [0], 101: [8], 128: [21, 29], 177: [2], 249: [0], 362: [0], 392: [5], 41
3: [1], 414: [0, 4], 455: [15], 494: [9], 549: [29], 587: [0, 3], 606: [13], 626: [3], 627: [2], 676: [19], 684: [1
9], 728: [4], 826: [0], 837: [0], 907: [3], 917: [2], 1101: [0], 1106: [10], 1250: [4], 1361: [4], 1369: [0], 1450:
[2], 1490: [4], 1556: [4], 1588: [1], 1746: [2], 1756: [38], 1820: [1, 4], 1997: [33], 1998: [11], 2061: [9], 2166:
[36], 2262: [16], 2288: [2], 2330: [0], 2331: [0], 2426: [19], 2551: [38], 2558: [0], 2563: [9], 2652: [3], 2661:
[4], 2669: [0], 2698: [5], 2718: [2], 2753: [17], 2777: [11], 2801: [0], 2931: [1], 2976: [14], 3067: [0], 3188:
[1], 3276: [12], 3286: [21], 3338: [3], 3371: [7], 3380: [2], 3450: [5], 3597: [0], 3637: [30], 3754: [1], 3994:
[1], 4119: [24], 4220: [1], 4381: [6], 4386: [21], 4501: [9], 4505: [10], 4531: [11], 4815: [17], 4835: [1], 5019:
[4, 7], 5074: [0], 5083: [6], 5184: [0], 5248: [8], 5259: [2], 5264: [7], 5291: [1], 5437: [2], 5533: [9], 5770:
[0, 2], 5930: [4, 11], 5931: [4, 11], 5948: [2], 5983: [3], 6151: [13], 6191: [4], 6277: [8], 6309: [0, 9], 6312:
[4], 6329: [5], 6363: [15], 6430: [3], 6447: [3], 6458: [2], 6565: [3], 6584: [4], 6627: [16], 6641: [17], 6654:
[9], 6719: [9], 6745: [3], 6763: [21], 6828: [11], 6872: [13], 6873: [3], 6897: [43], 6899: [16], 6906: [6], 6908:
[2], 6985: [4], 7007: [3], 7041: [9], 7157: [11], 7166: [9], 7184: [19], 7220: [14], 7238: [0], 7289: [1], 7477: [1
9], 7496: [12], 7519: [18], 7524: [4], 7554: [8], 7560: [1], 7650: [1], 7721: [0], 7782: [20], 7828: [7], 7838:
[1], 7961: [0], 8017: [0], 8036: [8], 8255: [14], 8434: [16], 8519: [5], 8615: [41], 8634: [22], 8853: [0], 9027:
[4], 9068: [4], 9322: [22, 29], 9335: [3], 9355: [3], 9395: [3], 9399: [3], 9456: [8], 9457: [8], 9556: [2], 9650:
[5], 9672: [0], 9688: [4], 9728: [3], 9763: [15], 9774: [15, 47], 9883: [13], 9981: [7], 10040: [11], 10064: [1, 4,
7], 10094: [3], 10214: [3], 10224: [8], 10230: [21, 24], 10381: [10], 10382: [3], 10383: [3], 10423: [23], 10433:
[13], 10454: [12], 10467: [4], 10471: [1], 10485: [2], 10603: [12], 10617: [2], 10699: [6], 10703: [2], 10819: [2
```

In [10]:
```
inv_index = {}
for pos,term in enumerate(data['stemmed_words']):
 for  word in term:
    if word in inv_index:
      if pos not in inv_index[word]:
        inv_index[word].append(pos)
    else:
      inv_index[word]=[]
      inv_index[word].append(pos)
for i in inv_index:
  print(i,":",inv_index[i], end='\n\n')
```

```
9, 8028, 8034, 8047, 8056, 8085, 8094, 8097, 8121, 8127, 8140, 8149, 8159, 8183, 8190, 8220, 8248, 8255, 8260, 827
4, 8295, 8299, 8329, 8330, 8335, 8346, 8361, 8382, 8386, 8398, 8401, 8402, 8418, 8422, 8423, 8427, 8432, 8445, 846
2, 8467, 8469, 8473, 8477, 8482, 8485, 8496, 8498, 8502, 8521, 8526, 8528, 8532, 8536, 8537, 8548, 8552, 8572, 858
3, 8586, 8587, 8591, 8611, 8624, 8626, 8628, 8633, 8648, 8649, 8652, 8655, 8660, 8666, 8675, 8678, 8683, 8690, 869
1, 8696, 8729, 8730, 8732, 8733, 8735, 8739, 8740, 8759, 8767, 8775, 8779, 8804, 8810, 8814, 8819, 8820, 8830, 883
1, 8845, 8849, 8858, 8860, 8866, 8868, 8872, 8873, 8884, 8886, 8914, 8947, 8963, 8969, 8974, 8980, 8984, 8986, 902
6, 9028, 9030, 9034, 9036, 9038, 9049, 9063, 9070, 9093, 9103, 9114, 9116, 9117, 9134, 9142, 9143, 9154, 9158, 917
4, 9177, 9179, 9185, 9194, 9221, 9223, 9233, 9235, 9236, 9260, 9263, 9264, 9274, 9294, 9301, 9302, 9307, 9337, 935
8, 9366, 9390, 9394, 9423, 9430, 9441, 9447, 9449, 9450, 9468, 9478, 9479, 9493, 9494, 9503, 9506, 9518, 9531, 954
0, 9547, 9551, 9552, 9555, 9556, 9565, 9566, 9569, 9570, 9579, 9581, 9588, 9595, 9596, 9609, 9619, 9631, 9638, 964
3, 9646, 9660, 9674, 9694, 9696, 9697, 9698, 9701, 9704, 9705, 9706, 9707, 9709, 9713, 9722, 9738, 9757, 9774, 979
0, 9811, 9813, 9814, 9834, 9840, 9849, 9856, 9859, 9865, 9866, 9868, 9907, 9911, 9912, 9920, 9923, 9926, 9928, 994
1, 9943, 9955, 9957, 9977, 9983, 10006, 10012, 10013, 10033, 10035, 10038, 10065, 10066, 10080, 10081, 10092, 1009
6, 10098, 10105, 10131, 10132, 10168, 10170, 10177, 10178, 10213, 10214, 10215, 10226, 10236, 10238, 10239, 10256,
10261, 10269, 10288, 10298, 10301, 10305, 10314, 10316, 10323, 10324, 10326, 10337, 10343, 10358, 10373, 10374, 103
76, 10378, 10387, 10388, 10391, 10392, 10399, 10404, 10408, 10423, 10424, 10428, 10433, 10438, 10449, 10459, 10476,
10480, 10491, 10497, 10500, 10502, 10505, 10516, 10519, 10523, 10526, 10531, 10535, 10537, 10543, 10550, 10551, 105
60, 10561, 10570, 10572, 10585, 10601, 10626, 10633, 10634, 10648, 10652, 10678, 10680, 10689, 10698, 10714, 10719,
10720, 10728, 10765, 10767, 10778, 10782, 10793, 10822, 10848, 10850, 10857, 10859, 10862, 10866, 10873, 10876, 108
```

In [12]:

```
query=input().lower().split(" ")
start_time=time.time()
inv_results=[]
if(len(query)>2):
    print("Invalid query")
else:
    try:
        inv_results=[value for value in inv_index[query[0]] if value in inv_index[query[1]]]
    except:
        print("No matches found")
    else:
        inv_results=set(inv_results)
        results=[]
        for res in inv_results:
            word1_poslist=pos_index[query[0]][1][res]
            word2_poslist=pos_index[query[1]][1][res]
            for i in word1_poslist:
                if (i+1 in word2_poslist):
                    results.append(res)
                    break
        for i in results:
            print(data['Title'][i]+'\n')
    finally:
        print("Query Execution time =", time.time()-start_time)
```

Outputs:

1. Mixture of upper case and lower case letters in query

```
        print("Query Execution time =", time.time()-start_time)
```

```
No matter
No matter who someone is, how they look like, what language they speak, what they wear, remember the human. For the s
ake of humanity, the working class can and must unite across all arbitrary boundaries.

Dems and corporate media keep saying "Vote blue no matter who" but that only applies until a socialist wins the democ
ratic primary, in which case they introduce an "independent" establishment candidate to make sure the blue doesn't wi
n.

This is the result of "Vote Blue no Matter Who".

Reminder: No matter what leftists worldwide disagree on, these are the 10 fundamental demands ALL socialists should a
lways agree on

“no matter which way elections go the pro-labour sentiment will not be reflected among the Democrats and Republicans.
This gives a glimpse, however, of the potential for a militant labour movement to build independent working-class pol
itical representation”

blue no matter who, amirite?!

Query Execution time = 0.0020034313201904297
```

2. All small case letters in query

```
look like
No matter who someone is, how they look like, what language they speak, what they wear, remember the huma
n. For the sake of humanity, the working class can and must unite across all arbitrary boundaries.

What Commitment to Diversifying the Federal Bench Looks Like — The president’s success in diversifying ou
r federal courts has gone underreported this year. It is our collective responsibility to champion this p
rogress and to call for its continuation

What Commitment to Diversifying the Federal Bench Looks Like — The president’s success in diversifying ou
r federal courts has gone underreported this year. It is our collective responsibility to champion this p
rogress and to call for its continuation

What Commitment to Diversifying the Federal Bench Looks Like — The president’s success in diversifying ou
r federal courts has gone underreported this year. It is our collective responsibility to champion this p
rogress and to call for its continuation

Looks like a comrade may be running the Kotaku Twitter account.

Bloomberg News Co-Founder Twists Numbers to Make MI Gov. Whitmer Look Like Economic Genius

How national elections will look like if you excluded votes from confederate states

What U.S. abortion access looks like, in graphics

Julie Bindel: The red light district of Amsterdam could soon be a distant memory | "What once looked like
a revolutionary approach to prostitution in Holland is now clearly seen as a disaster, by all except thos
e who seek to make a profit"

What Winning Looks Like

Rebekah Jones, fired Florida employee receives whistleblower status. Governor was lying about the COVID c
ases in the entire state of FL to make himself look like he was doing an amazing job. Ms Jones was trashe
d in the media by her employer and had a police raid on her home with guns drawn.

What does American illiberal democracy look like?

Looks like I will be Boycotting Kelloggs products

This is what accountability looks like: Ex-Cop Kim Potter sentenced to 24 months in Daunte Wright killin
g.

Looks like Former Trumpist Tom Brady got Fed Up with Trumps “Big Lie”

CNN Airs HUGE Explosion In the Sky Over Kyiv: 'It Looks Like a Large Aircraft'

What will an upcoming peak of the Red Scare look like and how concerned should we be?

YouGov - What do Americans think socialism looks like?

Trump and Mr Pillow Don't care about the 2020 election anymore. Maybe what they're trying to do is convin
ce enough people that it will be okay to try and steal the 2024 election. Lindell's whole argument looks
like nonsense to us. Maybe it's a diversion.

What Does Trans-Inclusive Abortion Advocacy Look Like?

Was reminded of the Movie “War Machine” and Will Poulter plays an Army Infantry soldier in Afghanistan. I
thought shit, he looks like a kid and that 18 year olds are not that far from being a kid. So I want to k
now, how many 18 year old Americans died in that bullshit war?

Query Execution time = 0.008003950119018555
```

3. Query length greater than 2

```
In [14]: query=input().lower().split(" ")
start_time=time.time()
inv_results=[]
if(len(query)>2):
    print("Invalid query")
else:
    try:
        inv_results=[value for value in inv_index[query[0]] if value in inv_index[query[1]]]
    except:
        print("No matches found")
    else:
        inv_results=set(inv_results)
        results=[]
        for res in inv_results:
            word1_poslist=pos_index[query[0]][1][res]
            word2_poslist=pos_index[query[1]][1][res]
            for i in word1_poslist:
                if (i+1 in word2_poslist):
                    results.append(res)
                    break
        for i in results:
            print(data['Title'][i]+'\n')
    finally:
        print("Query Execution time =", time.time()-start_time)
```

```
No matter what
Invalid query
```

Link to dataset:

https://www.kaggle.com/datasets/neelgajare/liberals-vs-conservatives-on-reddit-13000-posts