PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100 Ft. Road, BSK III Stage, Bengaluru – 560 085
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SESSION: AUG-DEC 2020

Course Title: Algorithms for Information Retrieval		
Course code: UE17CS412		
Semester : VII sem	**Section:G**	**Team Id:3**
SRN:PES1201701581	**Name:RAVICHANDRA GIDD**	
SRN:PES1201701635	**Name:AMBARISH D Y**	
SRN:PES1201701626	**Name:VENKATESH K**	
SRN:PES1201700239	**Name:PAVAN MITRA**	

ASSIGNMENT REPORT

Problem Statement
Search Engine based on a Vector-Space Model and a comparative evaluation alongside ElasticSearch Engine.

Description
Our 4 stage comparative study includes:

1. **Pre-processing:**
   - **We use the famed NLTK module to implement the following pre-processing techniques**

   - **Tokenization**

     It is the transformation of a bunch of meaningful data into a string of abstract characters that can be mapped to, later on.
     1. This helps in bringing in the abstraction of **term-id** which can later be used **not only** for more efficient querying **but also**
     2. Correction measures such as edit distance measurement for the sake of error correction.

- **Lemmatization**

  The process of reducing a term to its “root” word

  Ex: standardise -> standard

  Another option could have been stemming, where the same outcome is done using certain rules such as “remove -ish” etc. but sometimes, especially when there are a lot of proper nouns involved, it may lead to semantic loss.

  Ex: “fetish” -> “fet”

  We prefer lemmatization over stemming here since stemming may lead to loss of meaningful queries, owing to journalism usually dealing with tons of “**people, places and things!**”

  We use NLTK’s built-in **WordNet lemmatizer** that is sufficiently large and tailored to the English language

- **Case Folding**

  Since case (lower or upper) does not make differ semantically for searching, we “fold” all root query words to a single type (here, all lower case)

2. **B-tree implementation:**

   We use OOBTRee to build tree-based dictionaries and posting lists that are mapped to each entry

- We use a unique <term-frequency, doc-id> structure to index the inverted indices
- This structure can be put to use to find similarity measures later on

3. **<Op>Closest word:**

We used edit distance(Levenshtein distance) to obtain the closest word in the vocabulary.

It makes use of Dynamic Programming, a paradigm that invests in **overlapping subproblems** with repeatable measurements/ operations that build on each other.

$$\operatorname{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \operatorname{lev}(\operatorname{tail}(a), \operatorname{tail}(b)) & \text{if } a[0] = b[0] \\ 1 + \min \begin{cases} \operatorname{lev}(\operatorname{tail}(a), b) \\ \operatorname{lev}(a, \operatorname{tail}(b)) \\ \operatorname{lev}(\operatorname{tail}(a), \operatorname{tail}(b)) \end{cases} & \text{otherwise.} \end{cases}$$

4. **Ranking process**

After applying similar pre-processing measures to the query as well, we begin the 4 stage ranking process

- **Create a Term frequency vector**
  Refer to the term frequency using the “files” OOBTree and keep track of each query term’s tf score.
- **Created a inverse frequency vector**
  Refer to the posting lists and use a tf-idf matrix to similarly obtain idf scores.
- **Create a tf-idf vector**
  Using the “**document-at-a-time**” strategy, we obtain tf-idf score vector by multiplication.
- **Finally cosine similarity for retrieving top k results**

  We look at a simple Euclidean distance to calculate similarity between document vectors and the query vector, but this leads to some blatant dissimilarities.

Since the document tf-idf vectors exist in the same dimensions of the query tf-idf vectors, a simple dot product helps obtain a vector-based similarity measure for the same.

Now, a normalised dot-product of these vectors is equivalent to a cosine similarity(since cosine theta decreases with theta, and the closer the query to the document, the lower the angle, and hence larger the cosine measure**)**

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

This gives a convenient metric between [0,1] that can be ranked for all documents and returned by orders of top-k i.e. top-10, top-20 etc.

Code Snippets

```
for f in files:
    tf[f[:-4]]=dict()
    with open("./TelevisionNews/"+f) as csvfile:
            readCSV=csv.reader(csvfile,delimiter=',')
            t = OOBTree()
            for ind,row in enumerate(readCSV):
                pos=0
                N=N+1
                tf[f[:-4]][ind]=dict()
                if(len(row)==7):
                    for j in range(len(row)):
                        if(j==6):
                            l=nltk.tokenize.WhitespaceTokenizer().tokenize(row[j])
                            l = [nltk.stem.WordNetLemmatizer().lemmatize(word) for word in l]
                            l=[k.lower() for k in l]

                            for k in l:
                                #ps.stem(k)
                                if(k in tf[f[:-4]][ind]):
                                    tf[f[:-4]][ind][k]=tf[f[:-4]][ind][k]+1
                                else:
                                    tf[f[:-4]][ind][k]=1
                                if(k in df):
                                    df[k]=df[k]+1
                                else:
                                    df[k]=1
                                if(t.has_key(k)):
                                    if( ind in t[k]):
                                        t[k][ind].append(pos)
                                        pos=pos+1
                                    else:
                                        ll=list()
                                        ll.append(pos)
                                        pos=pos+1
                                        t[k][ind]=ll
                                else:
                                    ll=list()
                                    ll.append(pos)
                                    pos=pos+1
                                    dd=dict()
                                    dd[ind]=ll
                                    t[k]=dd
                        else:
                            #ps.stem(row[j])
                            if(row[j] in tf[f[:-4]][ind]):
                                tf[f[:-4]][ind][row[j]]=tf[f[:-4]][ind][row[j]]+1
                            else:
                                tf[f[:-4]][ind][row[j]]=1
                            if(row[j] in df):
                                df[row[j]]=df[row[j]]+1
                            else:
                                df[row[j]]=1
                            if(t.has_key(row[j])):
```

Open ▾ *air.py ~/AIR Save

search.py × *air.py ×

```python
                                    dd=dict()
                                    dd[ind]=ll
                                    t[k]=dd
                        else:
                            #ps.stem(row[j])
                            if(row[j] in tf[f[:-4]][ind]):
                                tf[f[:-4]][ind][row[j]]=tf[f[:-4]][ind][row[j]]+1
                            else:
                                tf[f[:-4]][ind][row[j]]=1
                            if(row[j] in df):
                                df[row[j]]=df[row[j]]+1
                            else:
                                df[row[j]]=1
                            if(t.has_key(row[j])):
                                if ind in t[row[j]]:
                                    t[row[j]][ind].append(pos)
                                    pos=pos+1
                                else:
                                    ll=list()
                                    ll.append(pos)
                                    pos=pos+1
                                    t[row[j]][ind]=ll
                            else:
                                ll=list()
                                ll.append(pos)
                                pos=pos+1
                                dd=dict()
                                dd[ind]=ll
                                t[row[j]]=dd
            afile = open(r"./"+f[:-4]+".pkl", 'wb')
            pickle.dump(t, afile)
            afile.close()
end=time.time()
df["N"]=N
afile = open(r"./"+"tf"+".pkl", 'wb')
pickle.dump(tf, afile)
afile.close()
afile = open(r"./"+"df"+".pkl", 'wb')
pickle.dump(df, afile)
afile.close()


file2 = open(r"df.pkl", 'rb')
new_d = pickle.load(file2)
file2.close()

file1 = open(r"df.pkl", 'rb')
df = pickle.load(file1)
file1.close()
file2 = open(r"tf.pkl", 'rb')
tf = pickle.load(file2)
file2.close()

```

Python ▾ Tab Width: 8 ▾ Ln 32, Col 10 ▾ INS

Open ▾ *air.py ~/AIR Save

search.py × *air.py ×

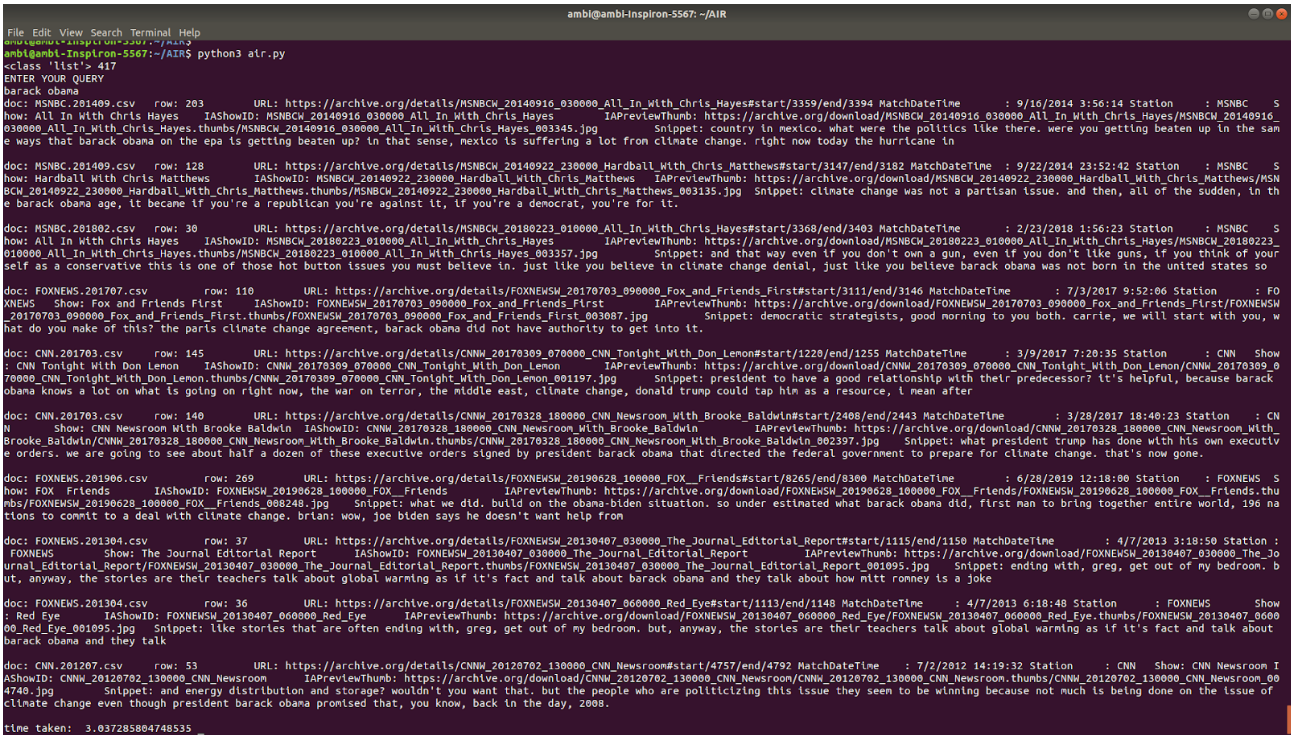
```python

def editDistDP(str1):
    if(str1 in df):
        return str1
    maxi=1000000000
    res=str1
    m=len(str1)
    for str2 in df:
        n=len(str2)
        dp = [[0 for x in range(n + 1)] for x in range(m + 1)]
        for i in range(m + 1):
            for j in range(n + 1):
                if(i == 0):
                    dp[i][j] = j
                elif(j==0):
                    dp[i][j] = i
                elif(str1[i-1] == str2[j-1]):
                    dp[i][j] = dp[i-1][j-1]
                else:
                    dp[i][j] = 1 + min(dp[i][j-1],dp[i-1][j],dp[i-1][j-1])
        if(dp[m][n]<maxi):
            res=str2
            maxi=dp[m][n]
        if(maxi==0):
            return res;
    return res
l=list()
#print(new_d["warming"],new_d["greenhouse"],new_d["gas"])
#query="greenhouses gasses and global warming "
query=input("ENTER YOUR QUERY\n")
start=time.time()
query=nltk.tokenize.WhitespaceTokenizer().tokenize(query)
query=[nltk.stem.WordNetLemmatizer().lemmatize(word) for word in query]

query=[i.lower() for i in query]
"""for i,q in enumerate(query):
    query[i]=editDistDP(q)"""
ql=[query.count(i)/len(query) for i in query]
for i in range(len(ql)):
    if(query[i] in df):
        ql[i]=ql[i]*math.log(df["N"]/df[query[i]])
    else:
        ql[i]=0
a=0
for i in ql:
    a=a+i*i
a=pow(a,0.5)
for f in files:
    for i,row in enumerate(tf[f[:-4]]):
        s=ntf(tf[f[:-4]][i])
        #print(s)
        ll=[0]*len(ql)
        for qi,q in enumerate(query):
```

Python ▾ Tab Width: 8 ▾ Ln 32, Col 10 ▾ INS

```python
        query[i]=editDistDP(q)"""
ql=[query.count(i)/len(query) for i in query]
for i in range(len(ql)):
        if(query[i] in df):
                ql[i]=ql[i]*math.log(df["N"]/df[query[i]])
        else:
                ql[i]=0
a=0
for i in ql:
        a=a+i*i
a=pow(a,0.5)
for f in files:
        for i,row in enumerate(tf[f[:-4]]):
                s=ntf(tf[f[:-4]][i])
                #print(s)
                ll=[0]*len(ql)
                for qi,q in enumerate(query):
                        if(q in tf[f[:-4]][i]):
                                ll[qi]=tf[f[:-4]][i][q]/s
                                ll[qi]*=math.log(df["N"]/df[q])
                b=0
                for j in ll:
                        b=b+j*j
                b=pow(b,0.5)
                if(b!=0):
                        s=0
                        for j in range(len(ql)):
                                s+=ql[j]*ll[j]
                        s=s/a
                        s=s/b
                        s=round(s,10)
                        res=list()
                        res.append(f)
                        res.append(i)
                        res.append(s)
                        l.append(res)
l.sort(key=lambda x:x[2])
l.reverse()

for i in range(10):
        if(i>=len(l)):
                print("no entries")
                break
        with open("./TelevisionNews/"+l[i][0]) as csvfile:
                readCSV=csv.reader(csvfile,delimiter=',')
                row=[i for i in readCSV]
                print(l[i])
                print("doc:",l[i][0],"\trow:",l[i][1],"\tURL:",row[l[i][1]][0],"MatchDateTime\t:",row[l[i][1]][1],"Station\t:",row[l[i][1]][2],"\tShow:",row[l[i][1]][3],"\tIAShowID:",row[l[i][1]][4],"\tIAPreviewThumb:",row[l[i][1]][5],"\tSnippet:",row[l[i][1]][6],"\n\n\n")

end=time.time()
print("time taken: ",end-start)
```

Output Screenshots

```
ambi@ambi-Inspiron-5567:~/AIR$ python3 air.py
<class 'list'> 417
grENTER YOUR QUERY
greenhouse gas and global warming
doc: BBCNEWS.201908.csv         row: 379        URL: https://archive.org/details/BBCNEWS_20190816_033000_HARDtalk#start/82/end/117 MatchDateTime        : 8/16/2019 3:31:37 Station     : BBCNEWS       Show
: HARDtalk      IAShowID: BBCNEWS_20190816_033000_HARDtalk      IAPreviewThumb: https://archive.org/download/BBCNEWS_20190816_033000_HARDtalk/BBCNEWS_20190816_033000_HARDtalk.thumbs/BBCNEWS_20190816_03300
0_HARDtalk_000058.jpg   Snippet: global greenhouse gas emissions are still rising. the data suggest the planet is warming at an alarming rate. what to do about it? well, my guest today is roger hallam, co
-founder of extinction rebellion, a movement dedicated to mass resistance and civil disobedience.

doc: BBCNEWS.201908.csv         row: 70         URL: https://archive.org/details/BBCNEWS_20190808_170000_BBC_News_at_Six#start/203/end/238 MatchDateTime        : 8/8/2019 17:03:38 Station     : BBCNEWS  S
how: BBC News at Six    IAShowID: BBCNEWS_20190808_170000_BBC_News_at_Six       IAPreviewThumb: https://archive.org/download/BBCNEWS_20190808_170000_BBC_News_at_Six/BBCNEWS_20190808_170000_BBC_News_at_Six
.thumbs/BBCNEWS_20190808_170000_BBC_News_at_Six_000178.jpg      Snippet: so what we choose to put an our plate helps define what the carbon footprint on the level of greenhouse gases in the atmosphere, so
 that choice between broccoli and ribs on your plate actually has a real link to the level of global warming that we're likely to see.

doc: FOXNEWS.201901.csv         row: 80         URL: https://archive.org/details/FOXNEWSW_20190115_050000_Tucker_Carlson_Tonight#start/3296/end/3331 MatchDateTime      : 1/15/2019 5:55:11 Station     : FO
XNEWS   Show: Tucker Carlson Tonight    IAShowID: FOXNEWSW_20190115_050000_Tucker_Carlson_Tonight       IAPreviewThumb: https://archive.org/download/FOXNEWSW_20190115_050000_Tucker_Carlson_Tonight/FOXNEWS
W_20190115_050000_Tucker_Carlson_Tonight.thumbs/FOXNEWSW_20190115_050000_Tucker_Carlson_Tonight_003297.jpg      Snippet: admitted that protesters have destroyed about 60% of the speed cameras in all of fr
ance. you might think global warming and speed cameras are not related, but they are. both our weapons are leaders use to bully the population. left to impose gas taxes on everyone else while flying

doc: BBCNEWS.201906.csv         row: 5  URL: https://archive.org/details/BBCNEWS_20190629_050000_Breakfast#start/2939/end/2974 MatchDateTime    : 6/29/2019 5:49:14 Station     : BBCNEWS       Show: Breakf
ast     IAShowID: BBCNEWS_20190629_050000_Breakfast     IAPreviewThumb: https://archive.org/download/BBCNEWS_20190629_050000_Breakfast/BBCNEWS_20190629_050000_Breakfast.thumbs/BBCNEWS_20190629_050000_Brea
kfast_002937.jpg        Snippet: because we were emitting carbon dioxide and methane into the air at a runaway rate, but what i hadn't fully understood is this - simply reducing greenhouse gas emissions w
ill not bring global warming under control.

doc: BBCNEWS.201906.csv         row: 4  URL: https://archive.org/details/BBCNEWS_20190630_033000_Click#start/119/end/154 MatchDateTime  : 6/30/2019 3:32:14 Station     : BBCNEWS       Show: Click     IASh
owID: BBCNEWS_20190630_033000_Click     IAPreviewThumb: https://archive.org/download/BBCNEWS_20190630_033000_Click/BBCNEWS_20190630_033000_Click.thumbs/BBCNEWS_20190630_033000_Click_000118.jpg        Snip
pet: dioxide and methane into the air at a runaway rate, but what i hadn't fully understood is this - simply reducing greenhouse gas emissions will not bring global warming under control.

doc: FOXNEWS.201203.csv         row: 70         URL: https://archive.org/details/FOXNEWSW_20120314_010000_Hannity#start/1786/end/1821 MatchDateTime     : 3/14/2012 1:30:01 Station     : FOXNEWS       Show
: Hannity       IAShowID: FOXNEWSW_20120314_010000_Hannity      IAPreviewThumb: https://archive.org/download/FOXNEWSW_20120314_010000_Hannity/FOXNEWSW_20120314_010000_Hannity.thumbs/FOXNEWSW_20120314_0100
00_Hannity_001765.jpg   Snippet: fuels and get the gasoline prices up to $9 dis. it hark back to those days? i think it does. we have moved on since then. global warming is not the headline that it was. i
nstead, we want a rational policy of getting gas prices under control, a rational energy

doc: BBCNEWS.201912.csv         row: 619        URL: https://archive.org/details/BBCNEWS_20191224_003000_Review_2019#start/717/end/752 MatchDateTime    : 12/24/2019 0:42:12 Station    : BBCNEWS       Show
: Review 2019   IAShowID: BBCNEWS_20191224_003000_Review_2019   IAPreviewThumb: https://archive.org/download/BBCNEWS_20191224_003000_Review_2019/BBCNEWS_20191224_003000_Review_2019.thumbs/BBCNEWS_20191224
_003000_Review_2019_000718.jpg  Snippet: that even sharp reductions in emissions won't, on their own, be enough to halt irreversible damage. so researchers here at cambridge university are looking at idea
s, however crazy, to try and take global warming gases out of the atmosphere

doc: BBCNEWS.201912.csv         row: 409        URL: https://archive.org/details/BBCNEWS_20191231_143000_Review_2019#start/771/end/806 MatchDateTime    : 12/31/2019 14:43:06 Station   : BBCNEWS       Show
: Review 2019   IAShowID: BBCNEWS_20191231_143000_Review_2019   IAPreviewThumb: https://archive.org/download/BBCNEWS_20191231_143000_Review_2019/BBCNEWS_20191231_143000_Review_2019.thumbs/BBCNEWS_20191231
_143000_Review_2019_000779.jpg  Snippet: global warming gases out of the atmosphere and to actually repair the earth's climate.

doc: BBCNEWS.201912.csv         row: 123        URL: https://archive.org/details/BBCNEWS_20191208_040000_BBC_News#start/1263/end/1298 MatchDateTime     : 12/8/2019 4:21:18 Station     : BBCNEWS       Show
: BBC News      IAShowID: BBCNEWS_20191208_040000_BBC_News      IAPreviewThumb: https://archive.org/download/BBCNEWS_20191208_040000_BBC_News/BBCNEWS_20191208_040000_BBC_News.thumbs/BBCNEWS_20191208_04000
0_BBC_News_001258.jpg   Snippet: greenhouse gas emissions. the findings suggest larger fish are being affected both by global warming and chemical pollution. tens of thousands of people are taking part in
 the world's big sleep out to raise awareness about homelessness. an estimated 100 million people across the globe don't have a home.

doc: BBCNEWS.201912.csv         row: 92         URL: https://archive.org/details/BBCNEWS_20191208_050000_BBC_News#start/351/end/386 MatchDateTime       : 12/8/2019 5:06:06 Station     : BBCNEWS       Show
: BBC News      IAShowID: BBCNEWS_20191208_050000_BBC_News      IAPreviewThumb: https://archive.org/download/BBCNEWS_20191208_050000_BBC_News/BBCNEWS_20191208_050000_BBC_News.thumbs/BBCNEWS_20191208_05000
0_BBC_News_000327.jpg   Snippet: which shows the earth's oceans are becoming starved of oxygen is further evidence that urgent action is needed to cut greenhouse gas emissions. the findings suggest larger
 fish are being affected both by global warming and chemical pollution. a state funeral has been

time taken:  3.6258487701416016
ambi@ambi-Inspiron-5567:~/AIR$
ambi@ambi-Inspiron-5567:~/AIR$
```



Interpretation of efficiency

Matches in the query	Time taken by custom search engine	Time taken by elastic search engine	Number of matches in top k	NAL score
greenhouse gas and global warming	3.6258	0.0109 s	12/18	1.13
barack obama	3.03728	0.0110 s	29/39	2.47
carbon dioxide	3.10845	0.0571 s	19/40	1.58

We observe that F scores are in the ranges of **[0.09, 0.15]**
This is insufficient as a comparative measure

Custom Metric:
We notice that as word count increases, the proportion of top-ks decreases, since ElasticSearch has complex context-based implementations, hence we penalise the lower word error more.

Normalised average log-depreciation top-k comparison (NAL score):
NAL = n(Custom U Es) / log (N)
, where N is the number of terms in query

It is also observed that the performance of professional IR systems is substantially higher for proper nouns (of the same length) which further leads us to believe they use complex cotext0based algorithms.

This leads to an idea of finding global reference index of a specific term and incorporating it into our measure as well (G-NAL score)

Learning Outcome

● Appreciating the nuances of a complex search engine
● Implementing tf.idf scoring, vector space ranking from scratch helps keep track of various data structures involved and their interconnectedness.
● Observed the importance of metrics and how specific measures can be used to seed insights from and tell stories about IR systems.

Name and Signature of the Faculty