20BCE204

IRS Practical 5

Consider a corpus of N documents. Implement Vector Space model (TFIDF consider normalized term frequency). Your implemented vector space model should rank the relevant retrieved documents by processing query.

In [71]:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import nltk
from nltk.corpus import stopwords
import math
```

In [90]:

```
docs=["Shipment of gold damaged in a fire","Delivery of silver arrived in a
      "Shipment of gold arrived in a truck"]

query=input("Enter the query :")

docs=[query]+docs
# docs.append(query)

docs
```

Out[90]:

```
Enter the query :gold silver truck
['gold silver truck',
 'Shipment of gold damaged in a fire',
 'Delivery of silver arrived in a silver truck',
 'Shipment of gold arrived in a truck']
```

In [91]:

```
from sklearn.feature_extraction.text import CountVectorizer
count_vectorizer = CountVectorizer()
bw = count_vectorizer.fit_transform(docs)
bw = pd.DataFrame(bw.toarray(),columns = count_vectorizer.get_feature_names

bw=bw.transpose()
bw
# print(vectorizer.get_feature_names())
# print(type(x))
```

Out[91]:

	0	1	2	3
arrived	0	0	1	1
damaged	0	1	0	0
delivery	0	0	1	0
fire	0	1	0	0
gold	1	1	0	1
in	0	1	1	1
of	0	1	1	1
shipment	0	1	0	1
silver	1	0	2	0
truck	1	0	1	1

In [92]:

```python
# # Get sum of all rows as a new row in Dataframe
# sum = bw[1:].sum()
# sum.name = 'Sum'
# # Assign sum of all rows of DataFrame as a new Row
# df = bw.append(sum.transpose())
# df.transpose()


dfi=[]
bw
for i in range(len(bw)):
    sum=0
    for j in range(1,len(docs)):
        if(bw.iloc[i,j]>=1):
            sum+=1
    dfi.append(sum)

bw['DF']=dfi
bw
```

Out[92]:

	0	1	2	3	DF
arrived	0	0	1	1	2
damaged	0	1	0	0	1
delivery	0	0	1	0	1
fire	0	1	0	0	1
gold	1	1	0	1	2
in	0	1	1	1	3
of	0	1	1	1	3
shipment	0	1	0	1	2
silver	1	0	2	0	1
truck	1	0	1	1	2

In [ ]:

```

```

In [93]:

```
n=len(docs)-1
n
idf=[]

for i in range(len(bw)):
    tdf=0
    idf.append(math.log10(n/bw['DF'][i]))
idf

bw['IDF']=idf
bw
```

Out[93]:

	0	1	2	3	DF	IDF
arrived	0	0	1	1	2	0.176091
damaged	0	1	0	0	1	0.477121
delivery	0	0	1	0	1	0.477121
fire	0	1	0	0	1	0.477121
gold	1	1	0	1	2	0.176091
in	0	1	1	1	3	0.000000
of	0	1	1	1	3	0.000000
shipment	0	1	0	1	2	0.176091
silver	1	0	2	0	1	0.477121
truck	1	0	1	1	2	0.176091

In [94]:

```
tfidf=[]
for i in range(len(docs)):
    q=np.array(bw.iloc[:,i]) * bw['IDF']
    bw[f"new {i}"]=q
bw
```

Out[94]:

	0	1	2	3	DF	IDF	new 0	new 1	new 2	new 3
arrived	0	0	1	1	2	0.176091	0.000000	0.000000	0.176091	0.176091
damaged	0	1	0	0	1	0.477121	0.000000	0.477121	0.000000	0.000000
delivery	0	0	1	0	1	0.477121	0.000000	0.000000	0.477121	0.000000
fire	0	1	0	0	1	0.477121	0.000000	0.477121	0.000000	0.000000
gold	1	1	0	1	2	0.176091	0.176091	0.176091	0.000000	0.176091
in	0	1	1	1	3	0.000000	0.000000	0.000000	0.000000	0.000000
of	0	1	1	1	3	0.000000	0.000000	0.000000	0.000000	0.000000
shipment	0	1	0	1	2	0.176091	0.000000	0.176091	0.000000	0.176091
silver	1	0	2	0	1	0.477121	0.477121	0.000000	0.954243	0.000000
truck	1	0	1	1	2	0.176091	0.176091	0.000000	0.176091	0.176091

In [106…

```
from scipy.spatial.distance import cosine
from pandas import DataFrame
cos=[]

for i in range(n):
    cos.append(1 - cosine(bw[f"new {i+1}"], bw["new 0"]))

cos
```

Out[106…

```
[0.08010451753994619, 0.8247514231034945, 0.32718457421366]
```