Name: Ezhilarasan C

Roll Number:225229151

Natural Language Processing Lab

Lab 3: Computing Documents Similarity using VSM

Exercise-1: Print TFIDF values

In [7]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
docs=[
    "good movie","not a good movie","did not like",
    "i like it","good one"]
tfidf=TfidfVectorizer(min_df=2 , max_df=0.5 ,ngram_range=(1,2))
features=tfidf.fit_transform(docs)
print(features)


```

```
  (0, 2)	0.7071067811865476
  (0, 0)	0.7071067811865476
  (1, 2)	0.5773502691896257
  (1, 0)	0.5773502691896257
  (1, 3)	0.5773502691896257
  (2, 3)	0.7071067811865476
  (2, 1)	0.7071067811865476
  (3, 1)	1.0
```

In [8]:
```python
df=pd.DataFrame(features.todense(),columns=tfidf.get_feature_names())
print(df)
```

```
   good movie      like     movie       not
0    0.707107  0.000000  0.707107  0.000000
1    0.577350  0.000000  0.577350  0.577350
2    0.000000  0.707107  0.000000  0.707107
3    0.000000  1.000000  0.000000  0.000000
4    0.000000  0.000000  0.000000  0.000000
```

Exercise-2:

1.Change the values of min_df and ngram_range and observe various output

In [9]:
```python
tfidf=TfidfVectorizer(min_df=1,max_df=.5,ngram_range=(2,4))
features=tfidf.fit_transform(docs)
print(features)
```

```
  (0, 2)	1.0
  (1, 2)	0.49552379079705033
  (1, 5)	0.6141889663426562
  (1, 6)	0.6141889663426562
  (2, 0)	0.5773502691896258
  (2, 7)	0.5773502691896258
  (2, 1)	0.5773502691896258
  (3, 4)	1.0
  (4, 3)	1.0
```

In [10]:
```
df=pd.DataFrame(features.todense(),columns=tfidf.get_feature_names())
print(df)
```

```
   did not  did not like  good movie  good one  like it  not good  \
0  0.00000       0.00000    1.000000       0.0      0.0  0.000000   
1  0.00000       0.00000    0.495524       0.0      0.0  0.614189   
2  0.57735       0.57735    0.000000       0.0      0.0  0.000000   
3  0.00000       0.00000    0.000000       0.0      1.0  0.000000   
4  0.00000       0.00000    0.000000       1.0      0.0  0.000000   

   not good movie  not like  
0        0.000000   0.00000  
1        0.614189   0.00000  
2        0.000000   0.57735  
3        0.000000   0.00000  
4        0.000000   0.00000  
```

Exercise-3: Compute Cosine Similarity between 2 Documents

In [11]:
```
from sklearn.metrics.pairwise import linear_kernel
```

In [12]:
```
doc1 = features[0:1]
doc2 = features[1:2]
score = linear_kernel(doc1, doc2)
print(score)
```

```
[[0.49552379]]
```

In [13]:
```
scores = linear_kernel(doc1, features)
print(scores)
```

```
[[1.         0.49552379 0.         0.         0.        ]]
```

In [14]:
```
query = "I like this good movie"
qfeature = tfidf.transform( [query])
score2 = linear_kernel(doc1, features)
print(score2)
```

```
[[1.         0.49552379 0.         0.         0.        ]]
```

Exercise-4: Find Top-N Similar Documents

1.Consider the following documents and compute TFIDF values

In [15]:
```
docs=["the house had a tiny little mouse",
"the cat saw the mouse",
"the mouse ran away from the house",
"the cat finally ate the mouse",
"the end of the mouse story" ]
```

2. Compute cosine similarity between 3rd document ("the mouse ran away from the house") with all other documents. Which is more similiar documents

In [16]:
```
tfidf = TfidfVectorizer(min_df=2, max_df=0.5, ngram_range=(1, 2))
features = tfidf.fit_transform(docs)
print(features)
```

```
  (0, 1)	0.7071067811865476
  (0, 3)	0.7071067811865476
  (1, 0)	0.7071067811865476
  (1, 2)	0.7071067811865476
  (2, 1)	0.7071067811865476
  (2, 3)	0.7071067811865476
  (3, 0)	0.7071067811865476
  (3, 2)	0.7071067811865476
```

In [18]:
```
doc1=features[0:3]
s=linear_kernel(doc1, features)
print(s)
```

```
[[1. 0. 1. 0. 0.]
 [0. 1. 0. 1. 0.]
 [1. 0. 1. 0. 0.]]
```

3. Find Top-2 similar documents for the 3rd document based on cosine similarity value ?

In [19]:
```
scores2 = linear_kernel(doc1, features)
print(scores2)
```

```
[[1. 0. 1. 0. 0.]
 [0. 1. 0. 1. 0.]
 [1. 0. 1. 0. 0.]]
```

In [ ]:
```

```