Name:Viviyan Richards W

Roll no:205229133

Lab3. Computing Document Similarity using VSM

EXERCISE-1: Print TFIDF values

In [1]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [2]:
```
import pandas as pd
```

In [3]:
```
docs = ["good movie", "not a good movie", "did not like", "i like it", "good one'
```

In [4]:
```
tfidf = TfidfVectorizer(min_df=2, max_df=0.5, ngram_range=(1, 2))
features = tfidf.fit_transform(docs)
print(features)
```

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

In [5]:
```
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 outputs

In [6]:
```
tfidf = TfidfVectorizer(min_df=1, max_df=0.6, ngram_range=(1, 2))
features = tfidf.fit_transform(docs)
print(features)
```

```
  (0, 3)	0.6098184563533858
  (0, 8)	0.6098184563533858
  (0, 2)	0.5062044059286201
  (1, 10)	0.5422255279709232
  (1, 9)	0.4374641418373903
  (1, 3)	0.4374641418373903
  (1, 8)	0.4374641418373903
  (1, 2)	0.36313475547801904
  (2, 11)	0.4821401170833009
  (2, 1)	0.4821401170833009
  (2, 6)	0.3889876106617681
  (2, 0)	0.4821401170833009
  (2, 9)	0.3889876106617681
  (3, 7)	0.6141889663426562
  (3, 5)	0.6141889663426562
  (3, 6)	0.49552379079705033
  (4, 4)	0.6390704413963749
  (4, 12)	0.6390704413963749
  (4, 2)	0.42799292268317357
```

In [7]:
```
df = pd.DataFrame(
 features.todense(),
 columns=tfidf.get_feature_names())
print(df)
```

```
       did  did not      good  good movie  good one        it      like  \
0  0.00000  0.00000  0.506204    0.609818   0.00000  0.000000  0.000000   
1  0.00000  0.00000  0.363135    0.437464   0.00000  0.000000  0.000000   
2  0.48214  0.48214  0.000000    0.000000   0.00000  0.000000  0.388988   
3  0.00000  0.00000  0.000000    0.000000   0.00000  0.614189  0.495524   
4  0.00000  0.00000  0.427993    0.000000   0.63907  0.000000  0.000000   

    like it     movie       not  not good  not like      one  
0  0.000000  0.609818  0.000000  0.000000   0.00000  0.00000  
1  0.000000  0.437464  0.437464  0.542226   0.00000  0.00000  
2  0.000000  0.000000  0.388988  0.000000   0.48214  0.00000  
3  0.614189  0.000000  0.000000  0.000000   0.00000  0.00000  
4  0.000000  0.000000  0.000000  0.000000   0.00000  0.63907  
```

EXERCISE-3: Compute Cosine Similarity between 2 Documents

In [8]:
```
from sklearn.metrics.pairwise import linear_kernel
```

In [9]:
```
doc1 = features[0:1]
doc2 = features[1:2]
score = linear_kernel(doc1, doc2)
print(score)
```

```
[[0.71736783]]
```

In [10]:
```
scores = linear_kernel(doc1, features)
print(scores)
```

```
[[1.         0.71736783 0.         0.         0.2166519 ]]
```

In [11]:
```
query = "I like this good movie"
qfeature = tfidf.transform([query])
scor = linear_kernel(doc1, features)
print(scor)
```

```
[[1.         0.71736783 0.         0.         0.2166519 ]]
```

EXERCISE-4: Find Top-N similar documents

Question-1. Consider the following documents and compute TFIDF values

In [12]:
```
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"
]
```

Question-2. Compute cosine similarity between 3rd document (“the mouse ran away from the house”) with all other documents. Which is the most similar document?

In [13]:
```
tfidf = TfidfVectorizer(min_df=2, max_df=0.5, ngram_range=(1, 2))
features = tfidf.fit_transform(docs)
print(features)
```

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

In [14]:
```
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.]]
```

In [15]:
```
scores2 = linear_kernel(doc1, features)
print(scores2)
```

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