**SURIYA_225229140**

# Lab-5 : Stemming and Lemmatization on Movie Dataset

```
In [18]:  from zipfile import ZipFile
          import glob
          import pandas as pd
          import nltk
          from sklearn.feature_extraction.text import TfidfVectorizer
          from sklearn.metrics.pairwise import linear_kernel
          from nltk.corpus import stopwords
          import warnings
          warnings.filterwarnings('ignore')
```

## EXERCISE-1

```
In [19]:  file_name = "movies.zip"
          with ZipFile(file_name, 'r') as zip:
              zip.printdir()
```

```
File Name                                          Modified             Size
movies/                                            2018-01-19 08:32:38     0
movies/12 Angry Men.txt                            2018-01-17 20:40:42  1007
movies/12 Years a Slave.txt                        2018-01-17 20:42:50  6451
movies/4 Months, 3 Weeks and 2 Days.txt            2018-01-17 20:37:10  1151
movies/All About Eve.txt                           2018-01-17 20:33:18  1346
movies/American Graffiti.txt                       2018-01-17 20:44:30  3417
movies/Boyhood.txt                                 2018-01-17 20:27:14  1970
movies/Casablanca.txt                              2018-01-17 20:26:26  1896
movies/Citizen Kane.txt                            2018-01-17 20:23:56  1483
movies/Gone with the Wind.txt                      2018-01-17 20:38:10  1318
movies/Hoop Dreams.txt                             2018-01-17 20:34:12  7909
movies/Manchester by the Sea.txt                   2018-01-17 20:40:06  3674
movies/Moonlight.txt                               2018-01-17 20:31:42  2323
movies/My Left Foot.txt                            2018-01-17 20:38:50  1115
movies/Pan's Labyrinth.txt                         2018-01-17 20:32:18  4431
movies/Psycho.txt                                  2018-01-17 20:34:46  3727
movies/Ran.txt                                     2018-01-17 20:43:48  2207
movies/Singin' in the Rain.txt                     2018-01-17 20:29:42   782
movies/Some Like It Hot.txt                        2018-01-17 20:35:40  7489
movies/The Godfather.txt                           2018-01-17 20:25:32  4293
movies/Three Colors Red.txt                        2018-01-17 20:28:22  2892
```

In [20]:
```python
nltk.download('punkt')
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\1mscdsa20\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\1mscdsa20\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [21]:
```python
from nltk.stem import PorterStemmer
ps = PorterStemmer()
tokenizer = nltk.tokenize.WhitespaceTokenizer()
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
from nltk.stem import LancasterStemmer
ls = LancasterStemmer()
```

In [24]:
```python
files = [file for file in glob.glob("movies/*")]
for file in files:
    with open(file, 'r', encoding='cp1252') as f:
        contents = f.readlines()
        print(contents)
        print("***********************************************************")
        print(" ")
```

```
["Lumet's origins as a director of teledrama may well be obvious here in his
first film, but there is no denying the suitability of his style - sweaty clo
se-ups, gritty monochrome 'realism', one-set claustrophobia - to his subject.
Scripted by Reginald Rose from his own teleplay, the story is pretty contrive
d - during a murder trial, one man's doubts about the accused's guilt gradual
ly overcome the rather less-than-democratic prejudices of the other eleven me
mbers of the jury - but the treatment is tense, lucid, and admirably economic
al. Fonda, though typecast as the bastion of liberalism, gives a nicely under
played performance, while Cobb, Marshall and Begley in particular are highly
effective in support. But what really transforms the piece from a rather talk
y demonstration that a man is innocent until proven guilty, is the consistent
ly taut, sweltering atmosphere, created largely by Boris Kaufman's excellent
camerawork. The result, however devoid of action, is a strangely realistic th
riller."]
***********************************************************

['There are movies to which the critical response lags far behind the emotion
al one. Two days after seeing 12 Years a Slave, British director Steve McQuee
n's adaptation of the 1853 memoir of a free black man kidnapped into slavery,
```

## A. How many sentences in each file?

```
In [25]: files = [file for file in glob.glob("movies/*")]
         for file in files:
             with open(file, 'r', encoding='cp1252') as f:
                 contents = f.readlines()
                 for row in contents:
                     sent_text = nltk.sent_tokenize(row)
                     print("sentence tokenize ", len(sent_text))
```

```
sentence tokenize  5
sentence tokenize  4
sentence tokenize  0
sentence tokenize  4
sentence tokenize  0
sentence tokenize  5
sentence tokenize  0
sentence tokenize  4
sentence tokenize  0
sentence tokenize  5
sentence tokenize  0
sentence tokenize  7
sentence tokenize  0
sentence tokenize  2
sentence tokenize  4
sentence tokenize  0
sentence tokenize  2
sentence tokenize  0
sentence tokenize  2
sentence tokenize  0
```

## B. How many tokens in each file?

```
In [29]:  files = [file for file in glob.glob("movies/*")]
          for file in files:
              with open(file, 'r', encoding='cp1252') as f:
                  contents = f.readlines()
                  for row1 in contents:
                      words = nltk.word_tokenize(row1)
                  print("word tokenize ", len(words))
```

```
word tokenize  181
word tokenize  119
word tokenize  20
word tokenize  276
word tokenize  9
word tokenize  70
word tokenize  49
word tokenize  98
word tokenize  242
word tokenize  67
word tokenize  131
word tokenize  157
word tokenize  69
word tokenize  66
word tokenize  39
word tokenize  25
word tokenize  50
word tokenize  208
word tokenize  100
word tokenize  569
```

## C. How many tokens excluding stop words in each file?

```
In [30]: files = [file for file in glob.glob("movies/*")]
         for file in files:
             with open(file, 'r', encoding='cp1252') as f:
                 contents = f.readlines()
                 filtered_sentence = [w for w in words if not w in stop_words]
                 print("stopwords ", len(filtered_sentence))
```

```
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
stopwords  365
```

## D. How many unique stems (ie., stemming) in each file? (Use PorterStemmer)

```
In [31]: def port_stemSentence(sentence):
             tokenizer = nltk.tokenize.WhitespaceTokenizer()
             tok = tokenizer.tokenize(sentence)
             filtered_sentence = [w for w in tok if not w in stop_words]
             stem_sentence = []
             for word in filtered_sentence:
                 stem_sentence.append(ps.stem(word))
             return len(stem_sentence)
```

```python
In [32]: files = [file for file in glob.glob("movies/*")]
         for file in files:
             with open(file, 'r', encoding='cp1252') as f:
                 contents = f.readline()
                 print("porter_stemming ")
                 print(port_stemSentence(contents))
```

```
porter_stemming
96
porter_stemming
83
porter_stemming
20
porter_stemming
138
porter_stemming
63
porter_stemming
64
porter_stemming
20
porter_stemming
51
porter_stemming
131
porter_stemming
27
porter_stemming
53
porter_stemming
87
porter_stemming
35
porter_stemming
93
porter_stemming
23
porter_stemming
34
porter_stemming
52
porter_stemming
38
porter_stemming
33
porter_stemming
282
```

## E. How many unique stems (ie., stemming) in each file? (Use LancasterStemmer)

In [33]:
```python
def lan_stemSentence(sentence):
    tokenizer = nltk.tokenize.WhitespaceTokenizer()
    tok = tokenizer.tokenize(sentence)
    filtered_sentence = [w for w in tok if not w in stop_words]
    stem_sentence = []
    for word in filtered_sentence:
        stem_sentence.append(ls.stem(word))
    return len(stem_sentence)
```

```python
In [34]: files = [file for file in glob.glob("movies/*")]
         for file in files:
             with open(file, 'r', encoding='cp1252') as f:
                 contents = f.readline()
                 print("lancaster_stemming ")
                 print(port_stemSentence(contents))
```

```
lancaster_stemming
96
lancaster_stemming
83
lancaster_stemming
20
lancaster_stemming
138
lancaster_stemming
63
lancaster_stemming
64
lancaster_stemming
20
lancaster_stemming
51
lancaster_stemming
131
lancaster_stemming
27
lancaster_stemming
53
lancaster_stemming
87
lancaster_stemming
35
lancaster_stemming
93
lancaster_stemming
23
lancaster_stemming
34
lancaster_stemming
52
lancaster_stemming
38
lancaster_stemming
33
lancaster_stemming
282
```

## F. How many unique words (ie., lemmatization) in each file? (Use WordNetLemmatizer)

```python
In [53]: import nltk
         nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\1mscdsa20\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[53]: True

```python
In [54]: def lemmSentence(sentence):
             tokenizer = nltk.tokenize.WhitespaceTokenizer()
             tok = tokenizer.tokenize(sentence)
             filtered_sentence = [w for w in tok if not w in stop_words]
             lemm_sentence = []
             for word in filtered_sentence:
                 lemm_sentence.append(lemmatizer.lemmatize(word))
             return len(lemm_sentence)
```

```python
In [55]: for file in files:
             with open(file, 'r', encoding='cp1252') as f:
                 contents = f.readline()
                 print("lemmatization ")
                 print(lemmSentence(contents))
```

```
lemmatization
96
lemmatization
83
lemmatization
20
lemmatization
138
lemmatization
63
lemmatization
64
lemmatization
20
lemmatization
51
lemmatization
131
lemmatization
27
lemmatization
53
lemmatization
87
lemmatization
35
lemmatization
93
lemmatization
23
lemmatization
34
lemmatization
52
lemmatization
38
lemmatization
33
lemmatization
282
```

## EXERCISE-2

## Step-1 For each movie:

## Tokenize terms and build list of tokens

```
In [56]:  tok = []
          for file in files:
              with open(file,'r',encoding='cp1252') as f:
                  contents = f.read()
                  let=tokenizer.tokenize(contents)
                  tok.append(let)
          tok
```

```
Out[56]:  [["Lumet's",
            'origins',
            'as',
            'a',
            'director',
            'of',
            'teledrama',
            'may',
            'well',
            'be',
            'obvious',
            'here',
            'in',
            'his',
            'first',
            'film,',
            'but',
            'there',
            'is',
```

## Find lemmatized words from the tokens

```
In [57]:  import nltk
          nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\1mscdsa20\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

```
Out[57]:  True
```

```
In [58]:  tok_lem =[]
          for i in tok:
              for j in i:
                  to_lem = lemmatizer.lemmatize(j)
                  tok_lem.append(to_lem)
          tok_lem
```

```
Out[58]:  ["Lumet's",
           'origin',
           'a',
           'a',
           'director',
           'of',
           'teledrama',
           'may',
           'well',
           'be',
           'obvious',
           'here',
           'in',
           'his',
           'first',
           'film,',
           'but',
           'there',
           'is',
           '  '
```

## Step-2

## Build Term-Document matrix using TfIdfVectorizer

```
In [59]: for file in files:
             with open(file,'r',encoding='cp1252') as f:
                 contents = f.read()
                 tok = tokenizer.tokenize(contents)
                 filtered_sentence = [w for w in tok if not w in stop_words]
                 tfidf = TfidfVectorizer(min_df=2,max_df=0.5,ngram_range=(1,2))
                 features = tfidf.fit_transform(filtered_sentence)
                 df = pd.DataFrame(features.todense(),columns=tfidf.get_feature_names())
                 print(df)
```

```
    man  one  rather
0   0.0  0.0     0.0
1   0.0  0.0     0.0
2   0.0  0.0     0.0
3   0.0  0.0     0.0
4   0.0  0.0     0.0
..  ...  ...     ...
91  0.0  0.0     0.0
92  0.0  0.0     0.0
93  0.0  0.0     0.0
94  0.0  0.0     0.0
95  0.0  0.0     0.0

[96 rows x 3 columns]
     12  all  almost  and  beautiful  black  but  children  comes  cotton  \
0   0.0  0.0     0.0  0.0        0.0    0.0  0.0       0.0    0.0     0.0
1   0.0  0.0     0.0  0.0        0.0    0.0  0.0       0.0    0.0     0.0
2   0.0  0.0     0.0  0.0        0.0    0.0  0.0       0.0    0.0     0.0
3   0.0  0.0     0.0  0.0        0.0    0.0  0.0       0.0    0.0     0.0
```

## Step-3

## Take vectors of any two movies and compute cosine similarity

In [60]:
```python
with open(files[5],'r',encoding='cp1252')as f:
    contents = f.read()
    tok = tokenizer.tokenize(contents)
    filtered_sentence = [w for w in tok if not w in stop_words]
    tfidf = TfidfVectorizer(min_df=2,max_df=0.5,ngram_range=(1,2))
    movie1 = tfidf.fit_transform(filtered_sentence)
    print(movie1)
```

```
(1, 10)        1.0
(5, 2)         1.0
(12, 13)       1.0
(15, 5)        1.0
(18, 10)       1.0
(31, 20)       1.0
(35, 12)       1.0
(37, 3)        1.0
(38, 9)        1.0
(45, 10)       1.0
(46, 11)       1.0
(48, 19)       1.0
(49, 16)       1.0
(53, 8)        1.0
(54, 4)        1.0
(56, 19)       1.0
(62, 20)       1.0
(65, 12)       1.0
(69, 7)        1.0
(72, 18)       0.5773502691896258
(72, 14)       0.5773502691896258
(72, 17)       0.5773502691896258
(77, 6)        1.0
(78, 18)       0.5773502691896258
(78, 14)       0.5773502691896258
  :       :
(108, 7)       1.0
(118, 5)       1.0
(121, 13)      1.0
(124, 12)      1.0
(128, 6)       1.0
(134, 10)      1.0
(138, 15)      1.0
(143, 15)      1.0
(148, 7)       1.0
(152, 1)       1.0
(154, 1)       1.0
(156, 1)       1.0
(165, 9)       1.0
(166, 0)       1.0
(172, 4)       1.0
(173, 2)       1.0
(174, 8)       1.0
(177, 10)      1.0
(179, 3)       1.0
(180, 0)       1.0
(188, 20)      1.0
(193, 7)       1.0
(194, 11)      1.0
```

```
(196, 12)      1.0
(203, 10)      1.0
```

```
In [61]: with open(files[10],'r',encoding='cp1252')as f:
             contents = f.read()
             tok = tokenizer.tokenize(contents)
             filtered_sentence = [w for w in tok if not w in stop_words]
             tfidf = TfidfVectorizer(min_df=2,max_df=0.5,ngram_range=(1,2))
             movie2 = tfidf.fit_transform(filtered_sentence)
             print(movie2)
```

```
  (0, 15)       1.0
  (1, 27)       1.0
  (2, 34)       1.0
  (3, 6)        1.0
  (4, 8)        1.0
  (7, 26)       1.0
  (11, 22)      1.0
  (13, 19)      1.0
  (15, 20)      1.0
  (17, 0)       1.0
  (29, 11)      1.0
  (34, 16)      1.0
  (46, 35)      1.0
  (52, 43)      1.0
  (53, 20)      1.0
  (62, 11)      1.0
  (66, 20)      1.0
  (67, 10)      1.0
  (71, 14)      1.0
  (73, 2)       1.0
  (74, 18)      1.0
  (77, 37)      1.0
  (78, 12)      1.0
  (81, 39)      1.0
  (82, 20)      1.0
  :        :
  (323, 34)     1.0
  (324, 25)     1.0
  (331, 42)     1.0
  (332, 19)     1.0
  (333, 40)     1.0
  (336, 23)     1.0
  (337, 29)     1.0
  (342, 31)     1.0
  (343, 33)     1.0
  (345, 38)     1.0
  (353, 3)      1.0
  (354, 11)     1.0
  (356, 24)     1.0
  (359, 28)     1.0
  (361, 27)     1.0
  (362, 34)     1.0
  (366, 43)     1.0
  (369, 22)     1.0
  (371, 30)     1.0
  (373, 41)     1.0
  (379, 4)      1.0
  (381, 36)     1.0
  (383, 7)      1.0
```

```
(384, 39)      1.0
(385, 4)       1.0
```

In [62]:
```python
doc1 = movie1[0:10]
doc2 = movie1[:]
score = linear_kernel(doc1,doc2)
print(score)
```

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

In [ ]: