In [1]:
```
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

The file movie.zip contains 20 files about various movies. For each of the files in movies.zip, you will have to do the following:

In [2]:
```
file_name = "movies.zip"            # opening the zip file in READ mode
with ZipFile(file_name, 'r') as zip:
    zip.printdir()                  # printing all the contents of the zip file
```

```
File Name                                             Modified             Size
movies/Three Colors Red.txt                    2021-05-04 04:18:04         2892
movies/The Godfather.txt                       2021-05-04 04:18:02         4293
movies/Some Like It Hot.txt                    2021-05-04 04:18:02         7489
movies/Ran.txt                                 2021-05-04 04:18:02         2207
movies/Psycho.txt                              2021-05-04 04:18:00         3727
movies/Pan_s Labyrinth.txt                     2021-05-04 04:18:00         4431
movies/My Left Foot.txt                        2021-05-04 04:17:58         1115
movies/Moonlight.txt                           2021-05-04 04:17:58         2323
movies/Manchester by the Sea.txt               2021-05-04 04:17:58         3674
movies/Hoop Dreams.txt                         2021-05-04 04:17:58         7909
movies/Citizen Kane.txt                        2021-05-04 04:17:56         1483
movies/Gone with the Wind.txt                  2021-05-04 04:17:56         1318
movies/Casablanca.txt                          2021-05-04 04:17:54         1896
movies/American Graffiti.txt                   2021-05-04 04:17:54         3417
movies/4 Months, 3 Weeks and 2 Days.txt        2021-05-04 04:17:52         1151
movies/All About Eve.txt                       2021-05-04 04:17:52         1346
movies/12 Angry Men.txt                        2021-05-04 04:17:52         1007
movies/12 Years a Slave.txt                    2021-05-04 04:17:52         6451
movies/Singin_ in the Rain.txt                 2021-05-04 04:18:02          782
```

In [3]:
```
files = [file for file in glob.glob("movies/*")]
files
```

Out[3]:
```
['movies\\12 Angry Men.txt',
 'movies\\12 Years a Slave.txt',
 'movies\\4 Months, 3 Weeks and 2 Days.txt',
 'movies\\All About Eve.txt',
 'movies\\American Graffiti.txt',
 'movies\\Boyhood.txt',
 'movies\\Casablanca.txt',
 'movies\\Citizen Kane.txt',
 'movies\\Gone with the Wind.txt',
 'movies\\Hoop Dreams.txt',
 'movies\\Manchester by the Sea.txt',
 'movies\\Moonlight.txt',
 'movies\\My Left Foot.txt',
 "movies\\Pan's Labyrinth.txt",
 'movies\\Psycho.txt',
 'movies\\Ran.txt',
 "movies\\Singin' in the Rain.txt",
 'movies\\Some Like It Hot.txt',
 'movies\\The Godfather.txt',
 'movies\\Three Colors Red.txt']
```

In [4]:
```
nltk.download('punkt')
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

```
[nltk_data] Error loading punkt: <urlopen error [Errno 11001]
[nltk_data]     getaddrinfo failed>
[nltk_data] Error loading stopwords: <urlopen error [Errno 11001]
[nltk_data]     getaddrinfo failed>
```

In [5]:
```
tokenizer = nltk.tokenize.WhitespaceTokenizer()
from nltk.stem import PorterStemmer
ps =PorterStemmer()
from nltk.stem import LancasterStemmer
ls = LancasterStemmer()
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

In [6]:
```
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?

B. How many tokens in each file?

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

In [7]:
```
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))
        for row1 in contents:
            words =nltk.word_tokenize(row1)
        print("word tokenize  ",len(words))
        filtered_sentence = [w for w in words if not w in stop_words]
        print("stopwords  ",len(filtered_sentence))
        print("***********")
```

```
sentence tokenize   5
word tokenize   181
stopwords   122
***********
sentence tokenize   2
word tokenize   119
stopwords   68
***********
sentence tokenize   1
word tokenize   20
stopwords   11
***********
sentence tokenize   7
word tokenize   276
stopwords   178
***********
sentence tokenize   1
word tokenize   9
stopwords   7
***********
```

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

In [8]:
```
def port_stemSentence(sentence):
    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)
```

In [9]:
```
for file in files:
    with open(file, 'r',encoding='cp1252') as f:
        contents = f.readline()
        print("porter_stemming  ")
        print(port_stemSentence(contents))
    print("********************")
```

```
porter_stemming
96
********************
porter_stemming
83
********************
porter_stemming
20
********************
porter_stemming
138
********************
porter_stemming
63
********************
porter_stemming
64
********************
porter_stemming
```

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

In [10]:
```
def lan_stemSentence(sentence):
    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)
```

In [11]:
```
for file in files:
    with open(file, 'r',encoding='cp1252') as f:
        contents = f.readline()
        print("lancaster_stemming  ")
        print(lan_stemSentence(contents))
    print("********************")
```

```
lancaster_stemming
96
********************
lancaster_stemming
83
********************
lancaster_stemming
20
********************
lancaster_stemming
138
********************
lancaster_stemming
63
********************
lancaster_stemming
64
********************
lancaster_stemming
```

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

In [12]:
```
def lemmSentence(sentence):
    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)
```

In [13]:
```
for file in files:
    with open(file, 'r',encoding='cp1252') as f:
        contents = f.readline()
        print("lemmatization ")
        print(lemmSentence(contents))
    print("********************")
```

```
lemmatization
96
********************
lemmatization
83
********************
lemmatization
20
********************
lemmatization
138
********************
lemmatization
63
********************
lemmatization
64
********************
lemmatization
```

EXERCISE-2

In this exercise, you will build your Term-Document Matrix for this movie collection of 20 movies. In order to improve the similarity search experience, you will use only lemmatized terms for creating the matrix.

Step-1 For each movie:

1. Tokenize terms and build list of tokens
2. Find lemmatized words from the tokens

In [14]:
```
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[14]:
```
[["Lumet's",
  'origins',
  'as',
  'a',
  'director',
  'of',
  'teledrama',
  'may',
  'well',
  'be',
  'obvious',
  'here',
  'in',
  'his',
  'first',
  'film,',
  'but',
  'there',
  'is',
  ' '
```

In [15]:
```
tok_lem =[]
for i in tok:
    for j in i:
        to_lem = lemmatizer.lemmatize(j)
        tok_lem.append(to_lem)
tok_lem
```

Out[15]:
```
["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 [16]:
```
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)
        print("******************")
```

```
    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
```

Step-3

Take vectors of any two movies and compute cosine similarity

In [17]:
```
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
```

In [18]:
```
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
```

In [19]:
```
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.]]
```