```
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')
```

Exe :1

```
file_name = "movies.zip"
with ZipFile(file_name, 'r') as zip:
    zip.printdir()
files = [file for file in glob.glob("movies/*")]
files.
nltk.download('punkt')
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

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()
```

Natural Language Processing Lab
Lab5. Stemming and Lemmatization on Movie Dataset

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:

A. How many sentences in each file?
B. How many tokens in each file?
C. How many tokens excluding stop words in each file?
D. How many unique stems (ie., stemming) in each file? (Use PorterStemmer)
E. How many unique stems (ie., stemming) in each file? (Use LancasterStemmer)
F. How many unique words (ie., lemmatization) in each file? (Use WordNetLemmatizer)
G. Pretty Printing: Print the details of A to E in the following order

File Name	Sentences	Tokens	Tokens-Only	StemsPorter	StemsLancaseter	Lemmas

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:
- Tokenize terms and build list of tokens
- Find lemmatized words from the tokens

Step-2
Build Term-Document matrix using TfIdfVectorizer

Step-3
Take vectors of any two movies and compute cosine similarity

EXERCISE-3

Will lemmatized matrix help to achieve better similarity search or not?. Please comment.

```
for file in files:
    with open (file, 'r', encoding='cp1252') as f:
        Contents = f.readlines()
        print (contents)
        print ("****************")
        print ("                ")
```

A. How many sentences in each file?

B. How many tokens in each file?

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

```
files = [files for file in glob.glob("movies/*")]
for file in files:
    with open (file, 'r', encoding='cp1252') as f:
     for row in contents:
         sent_text = nltk.sent_tokenize(row)
    print("sentence tokenize", len(sent_text))
    for row in contents
         words = nltk.word_tokenize(row1)
    print ("Word tokenize", len(words))
    filtered_sentences = [w for w in words if not w
                              in stop_words]
    print ("Stopwords", len(filtered_sentences))
    print ("*****")
```

D)

```
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)
```

NOTES

```
for files in files:
    with open (file, 'r', encoding = 'cp1252') as f:
        contents = f.readline()
        print ("porter_stemming")
    print (port_stem sentence (contents))
    print ("* * * * * * * * * *")
E). for lan_stem sentence (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 lan (stem_sentence)
for file in files:
    with open (file, 'r', encoding = 'cp1252') as f:
        contents = f.readline()
    print ("lancaster_stemming")
    print (lan_stem sentence (contents))
    print ("* * * * * * * * * *")
F) def lemmsentence (sentence):
        tok = tokenizer. tokenize (sentence).
    filtered_sentence = [w for w in tok if not w
                         in stop_words]
    lemm_sentences = []
    for word in filtered. sentence:
        lemm_sentence. append (lemmatizer. lemmatize
                               (word))
    return lan (lemm_sentence)
```

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

Exercise-2

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

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

Step:2

```
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("**********")
```

NOTES

Step 3:-

```
with open(file1.txt, '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)

with open(files['i05', '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)

doc1 = movie1[0:10]
doc2 = movie2[:]
score = linear_kernel(doc1, doc2)
print(score)
```