REPORT ON MOVIE REVIEWS

Team Name: Explorerz

ABSTRACT:

Movie reviews are an important way to gauge the performance of a movie. The entertainment industry requires new and better way to target specific users with certain features.

The main aim of this project is to identify the underlying sentiment of a movie review on the basis of its textual information. In this project, we try to classify whether a person liked the movie or not based on the review they give for the movie.

This project is exploring the possibility of classifying the movie review corpus as "positive" and "negative" to help enable make better decisions for target user, using data mining techniques.

REQUIREMENTS TO IMPLEMENT THE PROJECT:

HARDWARE:

1)Operating System-WINDOWS/LINUX(UBUNTU)

2)Minimum 4GB RAM

3)500GB Hard disk

SOFTWARE:

1)PYTHON: http://www.python.org/downloads/

2)NUMPY: http://sourceforge.net/projects/numpy/files/Numpy/

3)NLTK: http://pypi.python.org/pypi/nltk

DESIGN

The project is based on text classification .In text classification, we are given a description d& X of a document, where X is the document space and a fixed set of classes C={c1,c2,....cn}.

Classes are also called categories or labels. typically, the document space X is some type of high-dimensional space and the classes are human defined for the needs of an application .We are given a training set of labeled documents or test D.

In this project the classes C={Positive, Negative} and the document space is movie reviews.

CODE

```
import os, sys

import nltk

import random

file_paths1=[]

file_paths=[]

count = {}


DIR = r"C:\Users\Admin\AppData\Local\Programs\Python\Python35\pos" #location of pos
folder

for root,directories,files in os.walk(DIR):   #root diectory and files in the given dir

   for filename in files:

      filepath=os.path.join(root,filename)   #Joins one or more path components

      file_paths.append(filepath)   #file_paths contains the whole pathname

all_words=[]

lnames=[]

lpos=[[[],'pos']] #list of list of feature words from pos

for p in file_paths:

   lnames=open(p,'r').read().split() #reads the file and splits it into smaller parts

   lpos.append([lnames,'pos']) #we append the small words with pos tag to lpos

   for w in lnames:

      all_words.append(w) #all_words contains all split words from pos files

#print(lpos[1])

      DIR1 = r"C:\Users\Admin\AppData\Local\Programs\Python\Python35\neg" #location of
neg folder
```

```
for root,directories,files in os.walk(DIR1):
   for filename in files:
      filepath1=os.path.join(root,filename)
      file_paths1.append(filepath1)

for q in file_paths1:
   lnames=open(q,'r').read().split()
   lpos.append([lnames,'neg'])
   for w in lnames:
       all_words.append(w)
#print(lpos[1])
random.shuffle(lpos)
#print(len(all_words))
#print(lpos)

word_features=list(all_words)[:2000] #take first 2000 most frequent words

#print(all_words)
def document_features(document): #checks whether each of these words is present in a given document
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    return features
```

featuresets=[(document_features(d),c) for (d,c) in lpos]

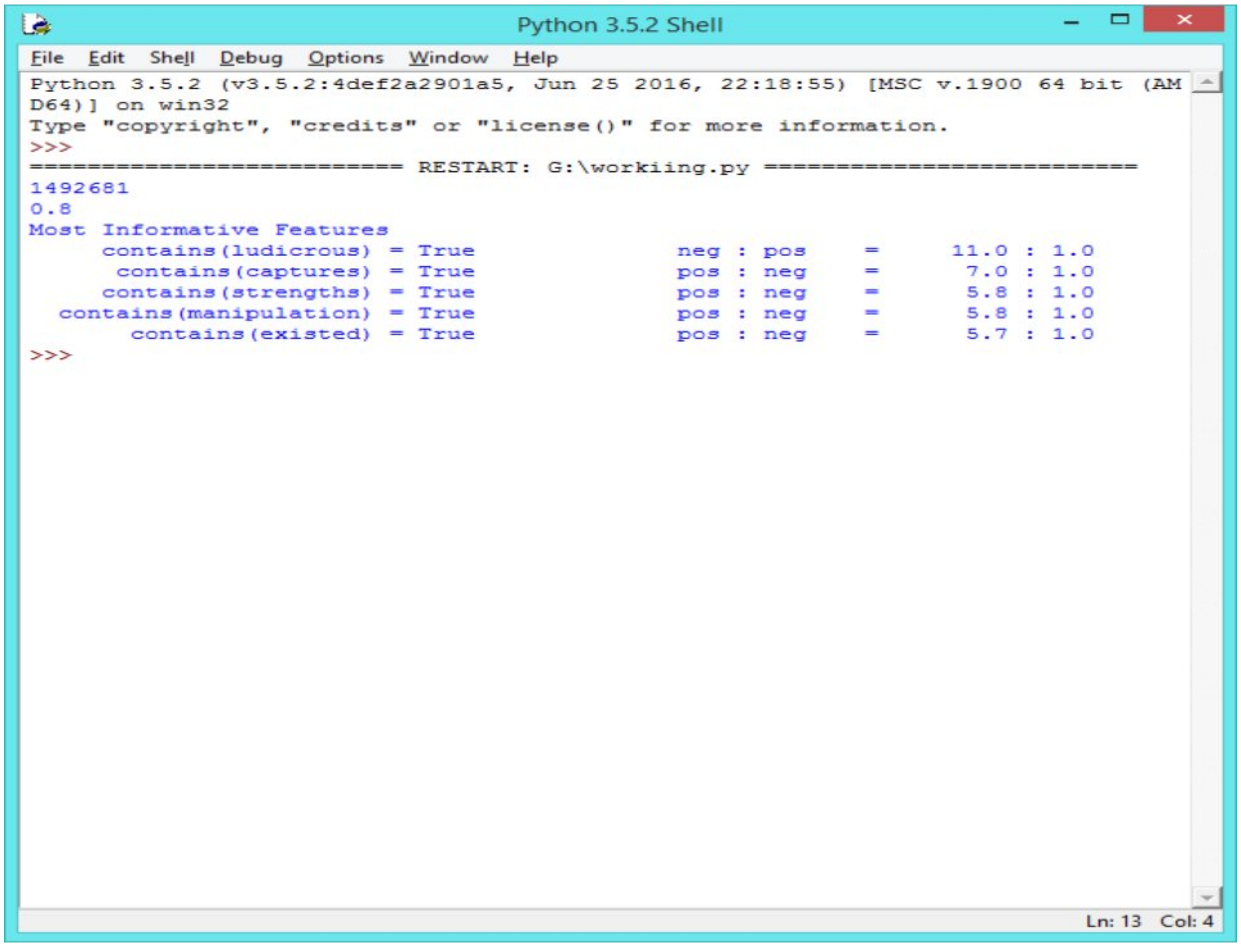
train_set,test_set=featuresets[100:],featuresets[:100] #splitting train set and test set

classifier = nltk.NaiveBayesClassifier.train(train_set) #train a classifier to label new movie reviews

print (nltk.classify.accuracy(classifier, test_set)) #printing accuracy

classifier.show_most_informative_features(5)   # features the classifier found to be most informative

<u>Output:</u>



```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========================== RESTART: G:\workiing.py ==========================
1492681
0.8
Most Informative Features
     contains(ludicrous) = True              neg : pos    =     11.0 : 1.0
      contains(captures) = True              pos : neg    =      7.0 : 1.0
     contains(strengths) = True              pos : neg    =      5.8 : 1.0
  contains(manipulation) = True              pos : neg    =      5.8 : 1.0
       contains(existed) = True              pos : neg    =      5.7 : 1.0
>>>
```

CONCLUSION AND CHALLENGES

One of the major improvements that can be incorporated as we move ahead in this project is to merge words with similar meanings before training the classifiers. Another point of improvement can be to mode this problem as a multi-class classification problem where we classify the sentiments of reviewer in more than binaryfashion like “Happy”, “Bored”, “Afraid”, etc. This problem can be further remodeled as a regression problem where we can predict the degree of affinity for the movie instead of complete like/dislike.

Working on this project has been a rewarding experience to us.

REFERENCES

- http://wiki.python.org/moin/BeginnersGuide
- https://docs.python.org/3/tutorial/
- https://developers.google.com/edu/python/