

MOVIE REVIEWS CLASSIFICATION

Group name : VISION

Team Members : Bhavana Gangadhar , Sri Raksha , Poorvi .K.N , Sushmitha .Y.S

ABSTRACT

This project is implementation of text classification. It explores the possibility of classifying the movie review corpus as positive or negative to make better decisions for target users. The sample data set given has files with movie review comments which are already tagged as positive or negative based on the review comments. A classifier is created using naïve bayes technique and is trained manually by a set of tagged training data. Accuracy of the model is obtained for new data set.

REQUIREMENTS

1. Extraction of features
2. Identifying the training, validation and test data
3. Creating and training the model using training data
4. Validating the model using validation data
5. Verifying the model with test data
6. Calculating the accuracy of classification

Hardware:

1. Operating System – Windows/Linux (Ubuntu)
2. Minimum 4 GB RAM
3. 500 GB Hard disk

Software:

4. Python <http://www.python.org/downloads/>
5. Numpy <http://sourceforge.net/projects/numpy/files/NumPy/>
6. NLTK: <http://pypi.python.org/pypi/nltk>

DESIGN

Classifier- In machine learning, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.

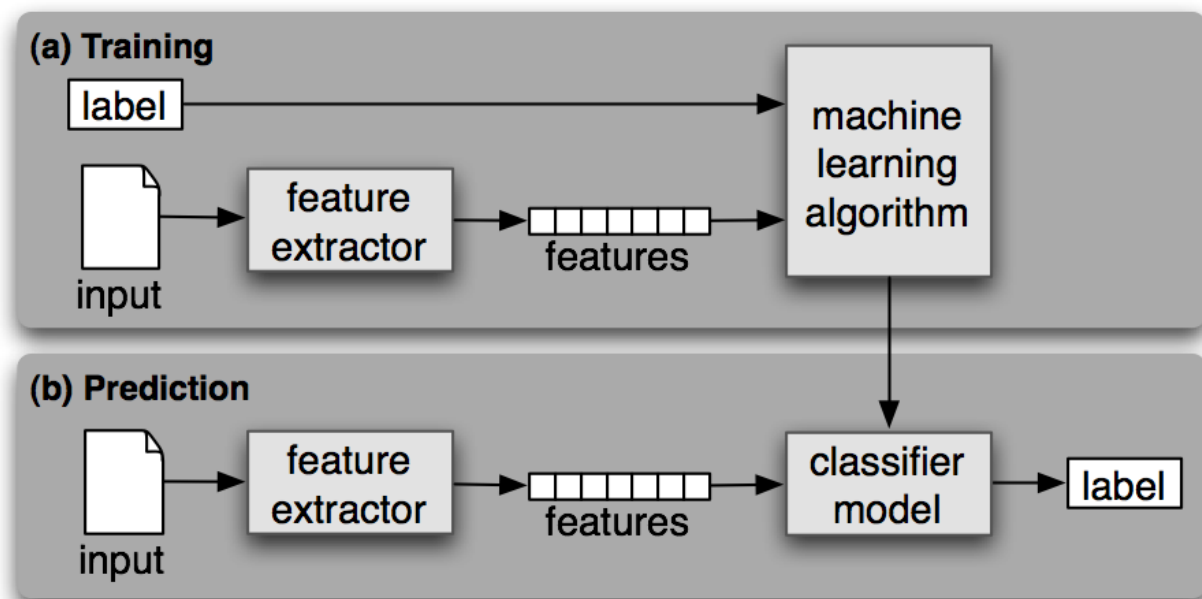


Figure shows Supervised Classification. (a) During training, a feature extractor is used to convert each input value to a feature set. These feature sets, which capture the basic information about each input that should be used to classify it, are discussed in the next section. Pairs of feature sets and labels are fed into the machine learning algorithm to generate a model. (b) During prediction, the same feature extractor is used to convert unseen inputs to feature sets. These feature sets are then fed into the model, which generates predicted labels.

Naive Bayes Classifier - It begins by calculating the prior probability of each label, which is determined by checking frequency of each label in the training set. The contribution from each feature is then combined with this prior probability, to arrive at a likelihood estimate for each label.

CODE

```
import nltk

import random

import os, sys

file_paths1=[]

file_paths=[]

DIR =r"C:\Users\Bhavana G\AppData\Local\Programs\Python\Python35-32\movie-reviews\pos"
#PATH TO THE POSITIVE REVIEWS

for root,directories,files in os.walk(DIR):

    for filename in files:

        filepath=os.path.join(root,filename)

        file_paths.append(filepath)


all_words=[]

lnames=[]

all_list=[[[],'pos']]

for p in file_paths:

    lnames=open(p,'r').read().split()

    all_list.append([lnames,'pos'])

    for w in lnames:

        all_words.append(w)


DIR1 = r"C:\Users\Bhavana G\AppData\Local\Programs\Python\Python35-32\movie-
reviews\neg" #PATH TO THE NEGATIVE REVIEWS

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()
    all_list.append([lnames,'neg'])
    for w in lnames:
        all_words.append(w)

random.shuffle(all_list)  #SHUFFLES BOTH THE POSITIVE AND NEGATIVE DATASETS

word_features = list(all_words)[:2000] #TOP MOST 2000 COMMON WORDS ARE
ASSIGNED TO word_features

def find_features(document):
    #FIND THESE TOP 2000 WORDS IN OUR POSITIVE AND NEGATIVE
    DOCUMENTS,MARKING THEIR PRESENCE AS EITHER POSITIVE OR NEGATIVE

    words = set(document)

    features = { }

    for w in word_features:
        features[w] = (w in words)

    return features

featuresets = [(find_features(rev), category) for (rev, category) in all_list]

training_set = featuresets[:1900] #TRAINING 1900 DATASET OUT OF 2000

testing_set = featuresets[1900:] #TESTING THE REMAINING 100 DATASET

classifier = nltk.NaiveBayesClassifier.train(training_set)

#TRAINING DATASET USING NAVIE BAYES CLASSIFIER

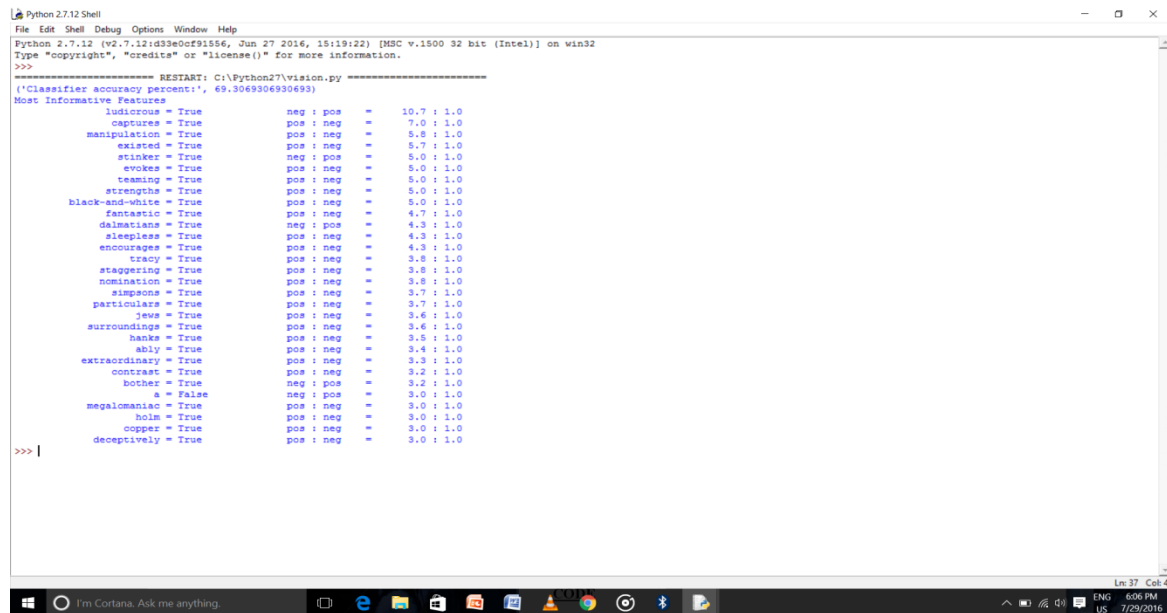
print("Classifier accuracy percent:",(nltk.classify.accuracy(classifier, testing_set))*100)

#TO PRINT ACCURACY PERCENT

classifier.show_most_informative_features(30) #TO LIST 30 MOST COMMON WORDS

```

OUTPUT SCREENSHOT



```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Python27\vision.py =====
('Classifier accuracy percent:', 69.3069306930693)
Most Informative Features
ludicrous = True          neg : pos = 10.7 : 1.0
captures = True          pos : neg = 7.0 : 1.0
manipulation = True       pos : neg = 5.8 : 1.0
existed = True           pos : neg = 5.7 : 1.0
stinker = True           neg : pos = 5.0 : 1.0
evokes = True            pos : neg = 5.0 : 1.0
teaming = True           pos : neg = 5.0 : 1.0
strengths = True         pos : neg = 5.0 : 1.0
black-and-white = True    pos : neg = 5.0 : 1.0
fantastic = True         pos : neg = 4.7 : 1.0
delusional = True        neg : pos = 4.3 : 1.0
sleepless = True         pos : neg = 4.3 : 1.0
encourages = True        pos : neg = 4.3 : 1.0
tiscy = True            pos : neg = 3.8 : 1.0
staggering = True        pos : neg = 3.8 : 1.0
nomination = True        pos : neg = 3.8 : 1.0
suspense = True          pos : neg = 3.7 : 1.0
particular = True        pos : neg = 3.7 : 1.0
jaws = True              pos : neg = 3.6 : 1.0
surroundings = True       pos : neg = 3.6 : 1.0
hanks = True             pos : neg = 3.5 : 1.0
ably = True              pos : neg = 3.4 : 1.0
extraordinary = True      pos : neg = 3.3 : 1.0
contrast = True          pos : neg = 3.2 : 1.0
bother = True            neg : pos = 3.2 : 1.0
a = False               neg : pos = 3.0 : 1.0
megalomaniac = True      pos : neg = 3.0 : 1.0
holm = True              pos : neg = 3.0 : 1.0
copper = True            pos : neg = 3.0 : 1.0
deceptively = True       pos : neg = 3.0 : 1.0
>>> |
```

CONCLUSION AND CHALLENGES

The project successfully enables in classifying the movie reviews files as positive or negative. The accuracy of classifier is checked using suitable functions of nltk. This accuracy depends on the number of words considered in the trained data.

The accuracy of the classifiers could be enhanced further if the common words are eliminated.

REFERENCES

- 1.<https://pythonprogramming.net>
- 2.<http://www.nltk.org/book/ch06.html>
- 3.<http://www.pythonforbeginners.com/files>