Heaven's light is our guide

Rajshahi University of Engineering & Technology, Bangladesh

Project Report

Course Name: Software Development Project II

Course No: CSE 3200

Prepared for

Mumu Aktar

Assistant Professor

Department of CSE

By

Md Zobair Hussain

ID: 143014, Session: 2014-15

Department of CSE

Introduction:

I. **Project Name:** Sentiment Analysis with Natural Language Processing.
II. **Technology Used:** NLTK (Natural Language tool kit), Python.
III. **Supported Operating System:**
This project can easily be run in Windows operating system. To run this project Spider of Anaconda python need to be installed.
IV. **Project Goals:**
a. Compare time complexity of different classifier for sentiment analysis.
b. To improve accuracy of different classifier by using updated dataset.
c. Compare accuracy among those classifiers.

V. **Project Objectives:**
Sentiment Analysis is one of the greatest applications of Computer Science. Sentiment Analysis is an application of Natural Language processing which is used to identify the opinion of a person about the product or discussing topic; whether it's negative, positive or neutral. It has a great importance in social media and business sector. In social media, it is used to measure the statistic. In business sector, marketing strategy of company, improvements of campaign success of a product, improvement of product messaging are greatly depending on it. Our task to develop a web based application to determine the sensitivity of the user input text which is as comment. There are many well-known techniques to solve this problem. As a beginner, I have chosen **Naive Bayes classifier.** This technique is easy to understand and implement. The accuracy of this technique is not very good but enough to work with it.

Sentiment Analysis:
Sentiment Analysis is the process of determining whether a piece of writing is positive, negative or neutral. It's also known as opinion mining, deriving the opinion or attitude of a speaker.

Naïve Bayes Classifier:
In this section, Naive Bayes classifier and its application to sentiment analysis, Text Parsing, Feature Vector are discussed.
Naive Bayes Classifier is enough powerful algorithm for classification task. It is based on Bayes Theorem. Bayes Theorem is a well-known in the field of probability. Bayes Theorem works on condition probability. Conditional

probability refers to the probability of happening of an event depends on the occurrence of another event which has already been.

If B1, B2, B3, ..., Bn are mutually exclusive events of sample space S where S = B1 ∪ B2 ∪ B3 ∪ ..... ∪ Bn

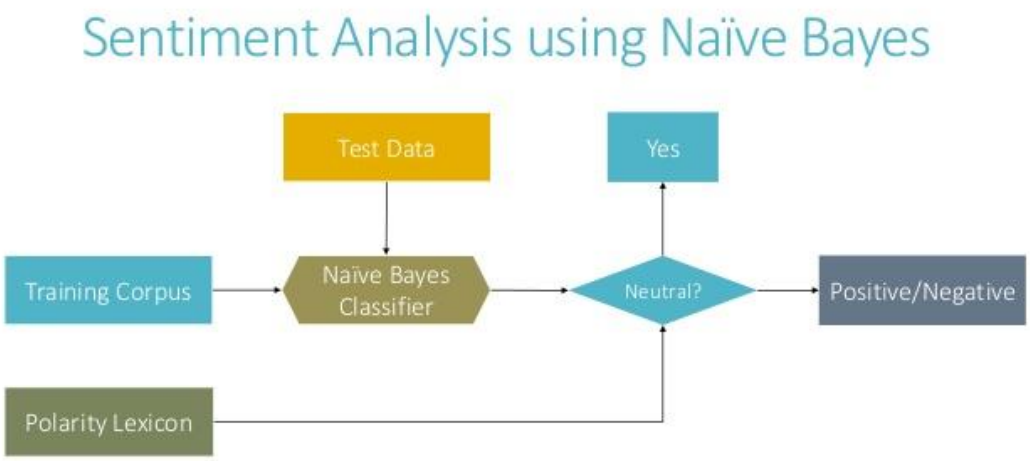
and P(Bi) > 0;

where i=1, 2, 3, ... ,n.

If A is another event of the sample space S such that P(A) > 0.

$$P(Bi|A) = P(Bi|A) = \frac{P(Bi)P(A|Bi)}{\sum_{i=1}^{n} P(Bi)P(A|Bi)}$$

where i=1,2,3,.....n. This is the Bayes Theorem.



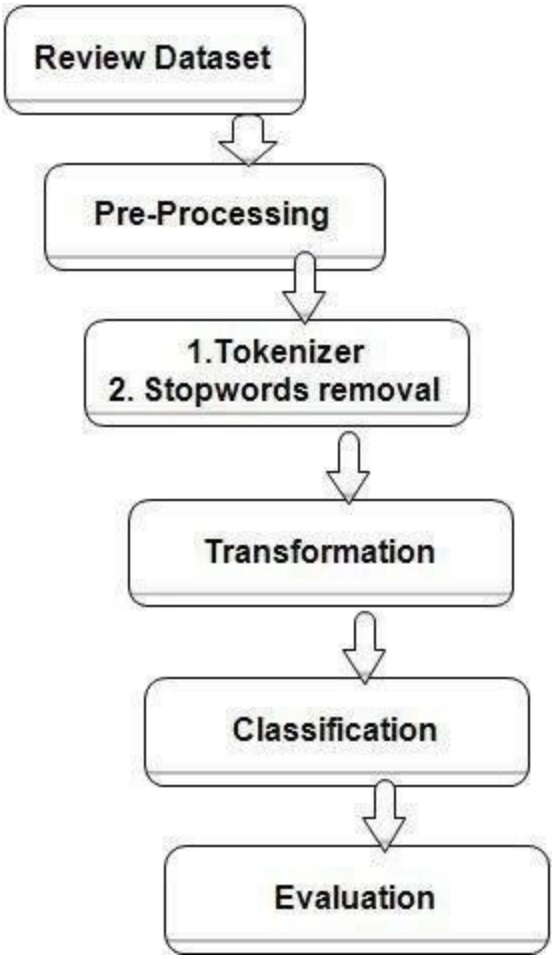
Data Set:

Data set is taken from Amazon short movie review.

Dataset link: http://jmcauley.ucsd.edu/data/amazon/

Evaluation metrics:

As a classification problem, Sentiment Analysis uses the evaluation metrics of Precision, Recall, F-score, and Accuracy. Also, average measures like macro, micro, and weighted F1-scores are useful for multi-class problems. Depending on the balance of classes of the dataset the most appropriate metric should be used.



K Fold Cross Validation:

Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it.

In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.

For classification problems, one typically uses stratified k-fold cross-validation, in which the folds are selected so that each fold contains roughly the same proportions of class labels.

In repeated cross-validation, the cross-validation procedure is repeated n times, yielding n random partitions of the original sample. The n results are again averaged (or otherwise combined) to produce a single estimation.

Code 01:

```
import nltk
import random
import pickle
from nltk.tokenize import word_tokenize

short_pos = open("short_reviews/positive.txt","r").read()
short_neg = open("short_reviews/negative.txt","r").read()

all_words = []
documents = []

allowed_word_types = ["J"]
for p in short_pos.split('\n'):
   documents.append( (p, "pos") )
   words = word_tokenize(p)
   pos = nltk.pos_tag(words)
   for w in pos:
      if w[1][0] in allowed_word_types:
         all_words.append(w[0].lower())

for p in short_neg.split('\n'):
   documents.append( (p, "neg") )
   words = word_tokenize(p)
   pos = nltk.pos_tag(words)
   for w in pos:
      if w[1][0] in allowed_word_types:
         all_words.append(w[0].lower())

save_documents = open("pickled_algos/documents.pickle","wb")
pickle.dump(documents, save_documents)
save_documents.close()
```

```
all_words = nltk.FreqDist(all_words)
word_features = list(all_words.keys())[:2000]

save_word_features = open("pickled_algos/word_features5k.pickle","wb")
pickle.dump(word_features, save_word_features)
save_word_features.close()

def find_features(document):
   words = word_tokenize(document)
   features = {}
   for w in word_features:
      features[w] = (w in words)
   return features

featuresets = [(find_features(rev), category)
            for (rev, category) in documents]
random.shuffle(featuresets)
save_featuresets = open("pickled_algos/featuresets5k.pickle","wb")
pickle.dump(featuresets, save_featuresets)
save_featuresets.close()
print(len(featuresets))
```

Code 02:

```
import pickle
import time
import nltk
import numpy as np
from sklearn.model_selection import KFold
from nltk.classify.scikitlearn import SklearnClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression

start_time = time.time()
documents_f = open("pickled_algos/documents.pickle", "rb")
documents = pickle.load(documents_f)
documents_f.close()
```

```
word_features5k_f = open("pickled_algos/word_features5k.pickle", "rb")
word_features = pickle.load(word_features5k_f)
word_features5k_f.close()

featuresets_f = open("pickled_algos/featuresets5k.pickle", "rb")
featuresets = pickle.load(featuresets_f)
featuresets_f.close()
print("--- %s seconds" % (time.time() - start_time))
start_time = time.time()

kf = KFold(n_splits=10)
sum = 0
n = 0
for train, test in kf.split(featuresets):
   train_data = np.array(featuresets)[train]
   test_data = np.array(featuresets)[test]
   LogisticRegression_classifier = SklearnClassifier(LogisticRegression())
   LogisticRegression_classifier.train(train_data)
   n=(nltk.classify.accuracy(LogisticRegression_classifier, test_data))
   print("Accuracy :",n)
   sum +=n
   print("--- %s seconds" % (time.time() - start_time))
   start_time = time.time()
average = sum/10
print("Average of LogisticRegretion: ",average*100)
print()

kf = KFold(n_splits=10)
sum = 0
n = 0
for train, test in kf.split(featuresets):
   train_data = np.array(featuresets)[train]
   test_data = np.array(featuresets)[test]
   MNB_classifier = SklearnClassifier(MultinomialNB())
   MNB_classifier.train(train_data)
   n=(nltk.classify.accuracy(MNB_classifier, test_data))
   print("Accuracy :",n)
   sum +=n
   print("--- %s seconds" % (time.time() - start_time))
   start_time = time.time()
```

```
average = sum/10
print("Average of    MNB_classifier: ",average*100)
print()
kf = KFold(n_splits=10)
sum = 0
n = 0
for train, test in kf.split(featuresets):
   train_data = np.array(featuresets)[train]
   test_data = np.array(featuresets)[test]
   classifier = nltk.NaiveBayesClassifier.train(train_data)
   n = nltk.classify.accuracy(classifier, test_data)
   print("Accuracy :",n)
   sum +=n
   print("--- %s seconds" % (time.time() - start_time))
   start_time = time.time()
average = sum/10
print("Average    of   NaiveBayes: ",average*100)
print()
```

Output:

runfile('D:/6th_semester/Project/Antora/Multinomial.py',

wdir='D:/6th_semester/Project/Antora')

--- 8.118993282318115 seconds

Accuracy : 0.6954076850984068

--- 67.05643320083618 seconds

Accuracy : 0.6850984067478912

--- 58.675769329071045 seconds

Accuracy : 0.6923076923076923

--- 55.77227783203125 seconds

Accuracy : 0.6866791744840526

--- 55.25835371017456 seconds

Accuracy : 0.6941838649155723

--- 57.7290780544281 seconds

Accuracy : 0.6894934333958724

--- 55.86580944061279 seconds

Accuracy : 0.699812382739212

--- 54.8404541015625 seconds

Accuracy : 0.6941838649155723

--- 55.772507429122925 seconds

Accuracy : 0.6857410881801126

--- 62.33784604072571 seconds

Accuracy : 0.7054409005628518

--- 59.276968479156494 seconds

Average of LogisticRegretion: 69.28348493347237

Accuracy : 0.6841611996251171

--- 61.86482048034668 seconds

Accuracy : 0.6504217432052484

--- 57.79116702079773 seconds

Accuracy : 0.6876172607879925

--- 55.82571244239807 seconds

Accuracy : 0.651031894934334

--- 55.10710072517395 seconds

Accuracy : 0.6378986866791745

--- 54.89905309677124 seconds

Accuracy : 0.6772983114446529

--- 59.46701216697693 seconds

Accuracy : 0.6575984990619137

--- 55.13922643661499 seconds

Accuracy : 0.6688555347091932

--- 54.8724000453949 seconds

Accuracy : 0.6772983114446529

--- 55.1662437915802 seconds

Accuracy : 0.7148217636022514

--- 55.09919857978821 seconds

Average of MNB_classifier: 67.0700320549453

Accuracy : 0.6982193064667291

--- 104.27632355690002 seconds

Accuracy : 0.7029053420805998

--- 103.1034185886383 seconds

Accuracy : 0.6960600375234521

--- 105.20397186279297 seconds

Accuracy : 0.6904315196998124

--- 105.28787779808044 seconds

Accuracy : 0.6951219512195121

--- 103.88590025901794 seconds

Accuracy : 0.6772983114446529

--- 105.4920666217804 seconds

Accuracy : 0.6960600375234521

--- 106.52275705337524 seconds

Accuracy : 0.6932457786116323

--- 104.18279027938843 seconds

Accuracy : 0.6829268292682927

--- 107.06216192245483 seconds

Accuracy : 0.7063789868667918

--- 105.60612845420837 seconds

Average of NaiveBayes: 69.38648100704927

Conclusion: From this project we can see that some classifier gives more accuracy and some less. But by seeing only accuracy we can't say which classifier is best. Because there are more factors like space complexity, time complexity etc. at the end, it can be said that, in the point of view of all attribute Naïve Bayes Classifier is better.