A REPORT

ON

ACTIVE LEARNING CLASSIFICATION OF SENTIMENT POLARITY OF MOVIE REVIEWS

By

Name of the student:	ID No.:
SAMISH BEDI	2012B3A7735P
RAM KARNANI	2012B3A7750P
LOKESH JAIN	2012B4A7827P

Prepared in the partial fulfillment of the course

MACHINE LEARNING
BITS F464

SUBMITTED TO
PROF. NAVNEET GOYAL

Department of Computer Science & Information Systems

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

November, 2015

TABLE OF CONTENTS

**Note: To see important code fragments go to Appendix A*

ACKNOWLEDGEMENT

An activity can be termed as an accomplishment only when its purpose is fulfilled. Its accomplishment involves a continuous unflinching effort, motivation and perennial support from its mentors and teammates. Course projects strengthens our theoretical, practical and technical concepts, which enhances our skills in the field of Computer Science. The successful realization of the project is the outgrowth of a consolidated effort of the people from disparate fronts.

Firstly, we would like to take this opportunity to thank, Prof. Navneet Goyal who provided us with the opportunity to work on this amazing project as well as constant guidance, overwhelming support and valuable teachings along the path of the research. His wisdom, clarity of thought and persistent encouragement motivated us to bring this project to its present state.

We would also like to thank almighty God for keeping us motivated throughout the project. The execution of various Data Mining and Machine Learning techniques and getting involved with the Active Learning approach helped us to understand the core concepts of this special case of semi-supervised learning. Last but not the least I am extremely thankful to all those who directly and indirectly helped me in completion of this endeavor.

1. Problem Statement and Assumptions:

In the Project at hand, the task is classification of Movie Reviews which requires the user to label each review as either "Positive" or "Negative". Having to manually annotate thousands of these instances can be tedious and even redundant. The key hypothesis is that if the learning algorithm is allowed to choose the data from which it learns - it will perform better with less training.

Our project looks specifically at the task of classification of **movie-reviews** labeled with respect to their overall ***sentiment polarity*** (positive or negative) into 2 sets of categories**.** The aim of this project is to find **optimal size of training set** to train the system and thereby making system classify the testing set with appreciable accuracy.
In past, substantial amount of work has been done on the classification of movie reviews but this work involves fairly large unlabeled data set. The major motivation behind the choice of project is to employ the principle of **Active Learning** and obtain an optimal dataset for training the model.

Our aim is to overcome the labeling bottleneck by asking queries in the form of unlabeled instances which are to be labeled by an oracle, usually a human annotator. For this project, we simulated a human annotator using an artificial labeler. In this way, the active learner aims to achieve high accuracy using as few labeled instances as possible, thereby minimizing the cost of obtaining labeled data.

We used following assumptions and methods for our purpose:

A. The reviews have been restricted for classification into 2 categories, i.e. **Positive** reviews and **Negative** reviews.
B. We are employing **Pool-Based Active Learning** approach to accomplish the aim of our project.
C. We are using **Query by Committee (QBC)** as the Query selection strategy where the classifiers employed for the purpose of labeling are allowed to vote on the labeling of query candidates.
D. To extract the optimal dataset for classification, Naive **Bayes Classification and K-Nearest Neighbours** techniques have been employed.
E. For the purpose of Naive Bayesian Classification we have taken into consideration the Conditional Independence Assumption i.e., the feature probabilities $P(x_i | c_j)$ are independent given the class c_j.

F. Bag of words Assumption i.e. Position of keyword occurrence within a review is of no significance.
G. In Naive Bayesian Classifier for calculating conditional probabilities of a class given document, we neglect probability of document which is same for each class
H. In Naive Bayesian Classifier we have used Laplace correction considering the fact that frequency of a keyword in a class can be zero.
I. In KNN, we have assumed the data to be in feature space, so as to define the notion of distance based similarity as well as cosine similarity.
J. Archive of classified reviews were classified randomly into training and testing groups.

Taking these assumptions into consideration the dataset is chosen to represent best possible scenario.

2. Description of Data and Source:

We extracted our dataset from Sentiment polarity datasets maintained by Cornell University. Our dataset constitutes of **10662** processed sentences/snippets, 5331 reviews from each of the two classes: Positive Reviews and Negative Reviews. This data has been taken from specifically from **Cornell Movie Review Data**, categorized under the subheading - Sentence Polarity Dataset v1.0. The original source of data is IMDB archive and arranged dataset extracted from Cornell University website has been annotated and preprocessed by using NLTK and python framework.

The dataset is then divided randomly in the ratio of **7:3 in training and testing set**. Using preprocessing techniques as mentioned in next section, we generated the vocabulary using the same.

3. Data Preprocessing:

We have used Sentiment Polarity dataset maintained by Cornell University for sentiment analysis and classification. The dataset has been categorized under 2 file heads. While composing tokens from each of the two file heads, the overlapping tokens from both the files have been eliminated.

For the purpose of preprocessing we have used text processing and Natural Language Processing techniques in order to generate meaningful tokens when training our model.

Our preprocessing method involved following steps:

A. **Sentence Parsing**: The sentiments of both categories of reviews have been processed and tokenized.
B. **Tokenization**: Tokens are generated from the dataset and stored into a list in python so that further operations can be applied in O(1) time. Let this dataset be K.
C. **Punctuations and special symbols are removed** from the dataset K and all the words which don't have follow up ASCII values are ignored
D. **Lemmatization and Stemming**: Technique of lemmatization to the keyword set obtained from Step(C) has been applied. Lemmatization has been done to remove inflectional endings only and to return the base or dictionary form of a word, thereby reducing the dimensions of our Model.
E. **Removal of stopwords**: New stopwords have been generated in addition to the list of stopwords already available with NLTK package, wherein those keywords which appear in more than 60% of the reviews have been regarded as fairly common, thereby non-differentiating and hence added as new stopwords. Tokens corresponding to these stopword and those already available with the NLTK package have been eliminated from our Keyword Set to reduce the probability of misclassification.

4. Active Learning Methodology Employed :

The method we employed for constructing the dataset is **Pool-Based Active Learning**. For many real-world learning problems, large collections of unlabeled data can be gathered at once. This motivates pool-based sampling, which assumes that there is a small set of labeled data L and a large pool of unlabeled data U available. Queries are selectively drawn from the pool, which we assume to be static in nature. We are querying the instances in a greedy fashion, according to an informativeness measure used to evaluate all instances in the pool.

For Query selection strategy, we have used **Query-By-Committee** approach which involves maintaining a committee $C = \{\theta(1), \ldots, \theta(C)\}$ of models which are all trained on the current labeled set L, but represent competing hypotheses. Each

committee member is then allowed to vote on the labeling of query candidates. The most informative query is considered to be the instance about which they most disagree. We are employing Naive Bayes Classifier and K- Nearest Neighbor for formulating our committee of classifier.

We constructed an artificial labeler to simulate a human labeler. Every time a review is annotated, we asked the artificial labeler to mark a word as a label for that reviewsWe allowed the labeler to return any one (and not necessarily the top one) of the positive words as a label for a positive review and any one of the negative words as a label for a negative review. To make this as practical as possible in a real-world setting, we constructed the artificial labeler to recognize only the most apparent words in the documents. However, in extreme cases, important unrecognized reviews can be displayed on terminal to query from actual human labeler.

In this special case of semi-supervised machine learning procedure where we employed the following algorithms to query the oracle to obtain the desired outputs at new data points.
Following are the important steps employed in the implementation:

1. We selected 100 positive sentiments and 100 negative sentiments from the labelled dataset file as initial training data.
2. Bayesian and KNN models are trained using this labeled data.
3. Both the trained models are employed to label the combined set of unlabeled data.

ALGORITHM: ACTIVE LEARNING

Input: U - unlabeled documents, L – labeled documents, θ - underlying classification models, B - budget
repeat
    $x^* = \underset{x \in U}{\operatorname{argmax}}\ QBC(x|\theta)$
    request label(y^*) from the Oracle
    $L \leftarrow L \cup \{(x^*, y^*)\}$
    $U \leftarrow U \setminus \{x^*\}$
    Train θ on L
until Budget B is exhausted; e.g., $|L| = B$

4. Query by Committee query selection strategy is used where we receive Classification Voting from our committee (Bayesian and KNN classifier) as the measure of disagreement.
5. We are aiming to minimize our version space of around 7000 sentiments to obtain optimal number of sentiments required for efficient training of our models.

6. The data points with disagreement in the committee are then queried from Oracle and assigned appropriated classes and added to training data for next iteration of Pool-based sampling.

   The classification models used by us for the project are:

   - **Naive Bayesian Classifier**: Naive Bayes is a simple technique for constructing classifiers and models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. We have assumed that the value of a particular feature is independent of the value of any other feature, given the class variable.
     We used the independence assumption of Naive Bayes classifier to calculate the probability of a document d being in class c as:

     $$P(c|d) \propto P(c) \prod_{1 \le k \le n_d} P(w_k|c)$$

     where $P(w_k|c)$ is the conditional probability of term w_k occurring in a document of class c and P(c) is the prior probability of a document occurring in class c. w_k is a token in d that is part of the vocabulary we use for classification and n_d is the number of such tokens in d. $P(w_i|c)$ is probability of a word w_i occurring in class c and is calculated using formula

     $$P(w_i|c_j)=(count(w_i,c_j)+1)/(\sum_{w \in V} count(w,c_j)+|V|)$$

     where $count(w_i|c_j)$ is the frequency of w_i in class c_j vocabulary. This probability is calculated as ratio of frequency of the word w_i in class c_j's vocabulary upon the sum of frequencies of all words in class c_j. For adjusting to the fact that frequency of a word can be 0 we have used Laplace Correction for which we add 1 to frequencies of all words in the vocabulary. For predicting class of a document we choose the class which has maximum probability given the document d:-

     $$c_{NB} = \text{argmax}_{c \in C} P(c) \prod_i P(w_i|c)$$

     where c_{NB} is the predicted class of naive bayesian classifier for the document d, C is the set of all classes and w_i is the set of tokens obtained from d which are part of the vocabulary extracted from the training set. In the end we also extended our model to Multi Label Classification to prove validity of our results.

   - **K Nearest Neighbor**- In *k-NN* classification an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically

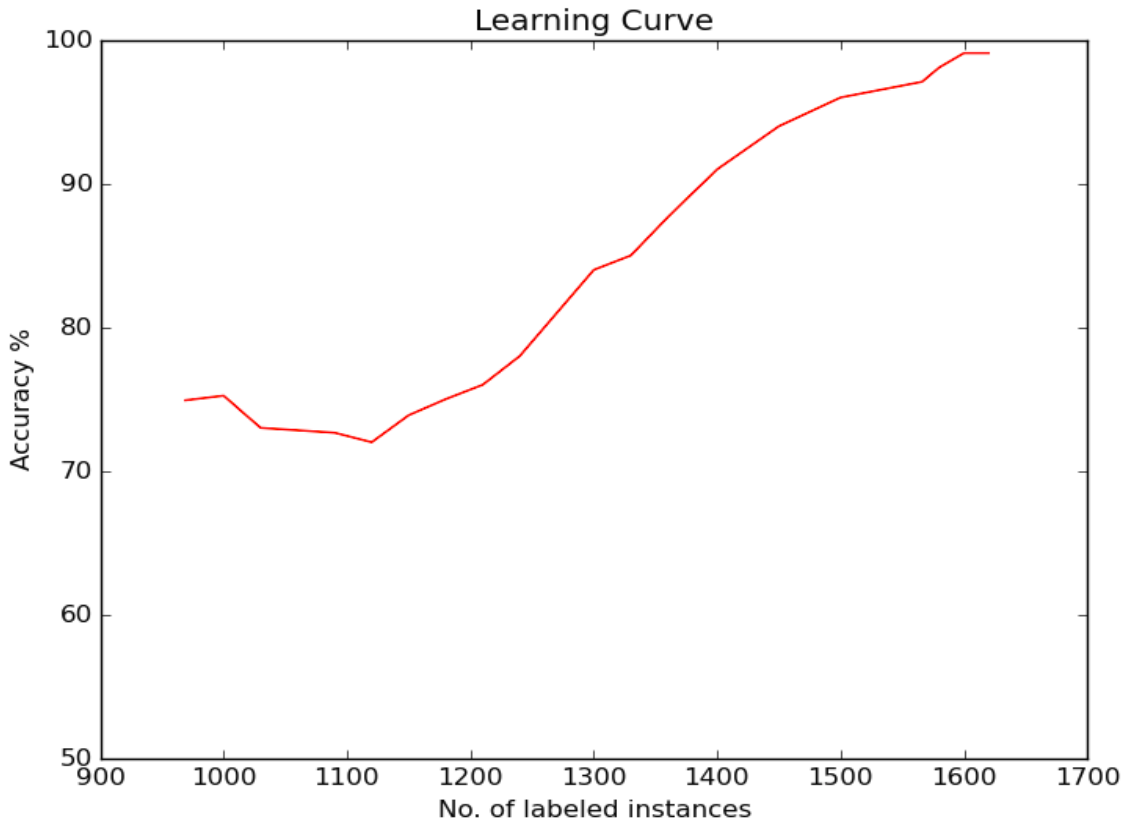
small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor. We have used k = 5 for our purpose. We tested our model for k=1, 3, 5, 7 and so on. For 1 and 3 we found that model was too simple and there was more generalization error. We first calculate class scores corresponding to each document and then select a class with maximum score. The class scores are calculated using formula:-

$$\mathbf{score(c,d) = \Sigma_{d' \in Sk(d)} I_c(d')\ cos(\vec{v}(d'),\vec{v}(d))}$$

where score(c,d) is the score of class c corresponding to document d, $S_k(d)$ is the set of d's k nearest neighbors and $I_c(d')$=1 iff d' is in class c and 0 otherwise. cos $(\vec{v}(d'),\vec{v}(d))$ corresponds to cosine similarity between vectors of d' and d.

5. Results and Analysis

In this section, we present the results of the project work by plotting the Learning Curve obtained during the training phase of our models using the approach of Active Learning. The table below highlights some of the notable points obtained on the learning curve.

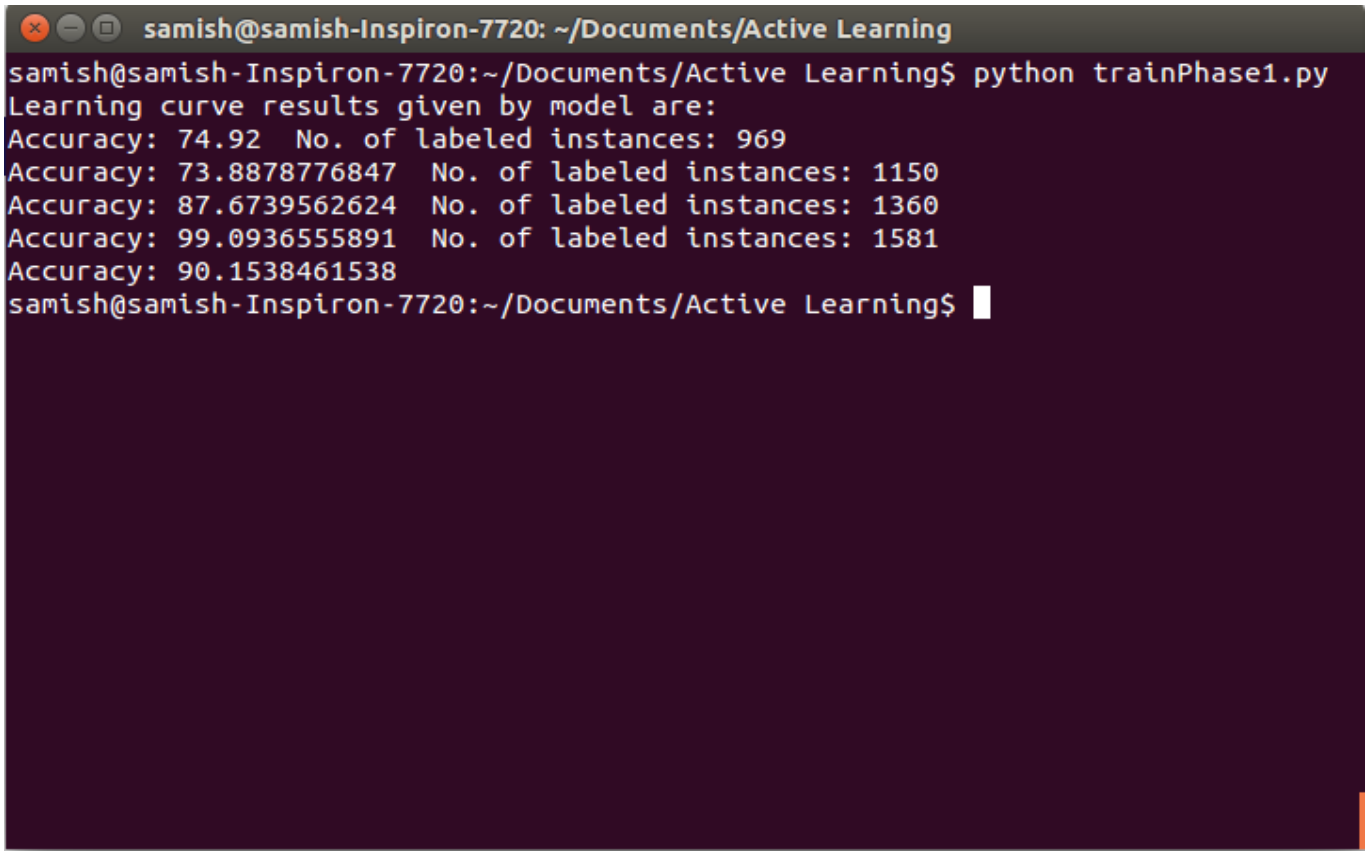


Learning curve Accuracy	Number of Instances
74.92	969
73.887	1150
87.673	1360
99.093	1581

Table 1: Learning curve accuracy corresponding to number of training instances

The learning curve depicts that we are able to efficiently train our models using 1581 data points instead of using the 7000 total data points. This was achieved by using the method of QBC where we were able to extract maximum information by finding the most informative query where our training model differed.

The trained model was run on the testing set of 3662 points and the model provided us with a **testing accuracy of 90.1538461538%** as compared to 71.2% accuracy of passive learning Naïve Bayesian algorithm with random sampling of training points. This validates the hypothesis that active learning strategy provided us with reduced training dataset to effectively train our model and can outperform traditional passive learning approach.



APPENDIX A: CODE:

*************************activeBayesianClassifier.py***************************************

Python module for Naive Bayesian Classifier

```
def bayesianClassifier(pos,neg,tokens):
	prob = [1.0,1.0]
	posl = len(pos)
	negl = len(neg)
	V = len(pos)+len(neg)
	checkp = False
	checkn = False
	for w in tokens :
		if(w in pos or w in neg):
			if w in neg:
				checkn = True
				prob[0] = prob[0]*(2.0/(negl+V))
				prob[1] = prob[1]*(1.0/(posl+V))
			else:
				checkp = True
				prob[0] = prob[0]*(1.0/(negl+V))
				prob[1] = prob[1]*(2.0/(posl+V))
	if(checkp==1 or checkn==1):
		prob.append(1)
	else:
		prob.append(0)
	return prob
```

*************************activeknn.py**

Python module for K-nearest neighbour classifier

```
def knnclassifier(posknn,negknn,token,k):
```

```
tokenl=len(token)
checkp = False
checkn = False
prob=[0.0,0.0]

posprob=[]
negprob=[]

for rev in posknn:
  posl = len(rev)
  freq=0.0
  for t in token:
     if(t in rev):
       checkp = True
       freq +=1
  if (posl>0 and tokenl>0):
    freq=freq/(posl*tokenl*1.0)
    posprob.append(freq)


for rev in negknn:
  negl = len(rev)
  freq=0.0
  for t in token:
    if(t in rev):
       checkn = True
       freq +=1
  if (negl>0 and tokenl>0):
    freq=freq/(negl*tokenl*1.0)
    negprob.append(freq)

posprob=sorted([w for w in posprob],reverse=True)
negprob=sorted([w for w in negprob],reverse=True)

posindex=0
negindex=0
#print checkp,checkn
```

```
 if(checkp==1 or checkn==1):
  i=0
  while(i<k):
   if(posprob[posindex]>negprob[negindex]):
    prob[1]=prob[1]+posprob[posindex]*1.0
    posindex +=1
   else:
    prob[0]=prob[0]+negprob[i]*1.0
    negindex+=1
   i+=1
  prob.append(1)
 else:
  prob.append(0)

 return prob
```

**************************run.py**

Python module for autorun file

```
 import nltkwordextraction
import activeBayesianClassifier as bayes
import os,sys
from random import randint
import wordextractor as we
from sets import Set
import activeknn as knn
import random
import pylab as pl
import matplotlib
import matplotlib.pyplot as plt
import numpy as np


f1 = open("Data/unlabeled.txt",'r')
f2 = open("Data/labeled.txt",'r')
```

```
poskeywords = []
negkeywords = []

f3 = open("posOr.txt",'r')
f4 = open("negOr.txt",'r')

for l in f3:
  poskeywords.append(l[:-1])
for l in f4:
  negkeywords.append(l[:-1])

knnpos = []
knnneg = []

pos = Set()
neg = Set()
maxp = 100
maxn = 100

supply=[]


unlabel= []
label = []


for l in f1:
  unlabel.append(l)

for l in f2:
  label.append(l)


var = []
def addPosKeywords(lp):
  tokens = we.make_token(lp)
```


```
  l = Set()
  for t in tokens:
      pos.add(t)
      l.add(t)
  if(len(l)!=0):
    knnpos.append(l)
  return l

def addNegKeywords(ln):
  tokens = we.make_token(ln)
  l = Set()
  for t in tokens:
      neg.add(t)
      l.add(t)
  if(len(l)!=0):
    knnneg.append(l)
  return l

train = []

index = 0
while(maxp>0):
  if label[index][0] == "1":
    train.append(label[index])
    maxp -=1
    tokens = we.make_token(label[index][1:])
    l = Set()
    for t in tokens:
      if t in poskeywords :
        pos.add(t)
        l.add(t)
    if(len(l)!=0):
      knnpos.append(l)
  index+=2

index = 1
while(maxn>0):
```

```
  if label[index][0] == "0":
    train.append(label[index])
    maxn -=1
    tokens = we.make_token(label[index][1:])
    l = Set()
    for t in tokens:
      if t in negkeywords :
        neg.add(t)
        l.add(t)
    if(len(l)!=0):
      knnneg.append(l)
    index+=2

extra = 0
dump = []
unlabelled=[]

for i in range(200,7000):
  unlabelled.append(label[i][1:])

xaxis=[]
yaxis=[]
itr=0
while(itr<4):
  itr+=1
  lp=[]
  ln=[]
  acc=0
  acc1=0
  accden=0

  for i in range(0,len(unlabelled)):
    if unlabelled[i]!="#":
      accden+=1
      active=0
      index = i
      tokens = we.make_token(unlabelled[index])
```

```
    bayesprob = bayes.bayesianClassifier(pos,neg,tokens.keys())
    knnprob = knn.knnclassifier(knnpos,knnneg,tokens.keys(),5)
    if((bayesprob[1]>bayesprob[0])==(label[index+200][0]=="1") or
(bayesprob[0]>bayesprob[1])==(label[index+200][0]=="0")):
      acc = acc+1
    if((knnprob[1]>knnprob[0])==(label[index+200][0]=="1") or
(knnprob[0]>knnprob[1])==(label[index+200][0]=="0")):
      acc1 = acc1+1


    if(knnprob[-1]==0 and bayesprob[-1]==0):
      abc=0
    else:
      if((bayesprob[1]>=bayesprob[0] and knnprob[1]<=knnprob[0]) or
(bayesprob[1]<=bayesprob[0] and knnprob[1]>=knnprob[0]) and (bayesprob[-1]==1 or
knnprob[-1]==1)):
        active=1
        if(label[index+200][0]=="1"):
          lp.append(unlabelled[index])
          train.append("1"+unlabelled[index])
          unlabelled[index]="#"
        else:
          ln.append(unlabelled[index])
          train.append("0"+unlabelled[index])
          unlabelled[index]="#"
      elif((bayesprob[1]>=bayesprob[0] and knnprob[1]>=knnprob[0]) or
(bayesprob[1]<=bayesprob[0] and knnprob[1]<=knnprob[0]) and bayesprob[-1]==1 and
knnprob[-1]==1):
        unlabelled[index]="#"

  yaxis.append((acc*1.0/accden*1.0)*100.0 )
  xaxis.append(len(train))

  for x in range(0,len(lp)):
    addPosKeywords(lp[x])
  for x in range(0,len(ln)):
    addNegKeywords(ln[x])
```

```
print "Learning curve results given by model are:"
for i in range(0,4):
    print "Accuracy:",yaxis[i]," No. of labeled instances:",xaxis[i]

plt.ylim(50,100)
plt.xlabel('No. of labeled instances')
plt.ylabel('Accuracy %')
plt.title('Learning Curve')
plt.plot(xaxis,yaxis,'r')
plt.show()

test=[]
testpos=0
testneg=0
acc=0
for i in range(7000,10662):
  if(label[i][0]=="1"):
    testpos+=1
  elif(label[i][0]=="0"):
    testneg+=1
  test.append(label[i])

lentot=len(test)
for i in range(0,len(test)):
  tokens = we.make_token(test[i][1:])
  knnprob = knn.knnclassifier(knnpos,knnneg,tokens.keys(),5)

  if((knnprob[1]>knnprob[0])==(label[i+10201][0]=="1") or
(knnprob[0]>knnprob[1])==(label[i+10201][0]=="0") and knnprob[-1]!=0):
    acc = acc+1

print "Accuracy:",((acc*1.0)/(lentot)*1.0)*100.0
```

APPENDIX B: REFERENCES:

1. Text Classification and Naive Bayes: The Task of Text Classification https://web.stanford.edu/class/cs124/lec/naivebayes.pdf
2. k Nearest Neighbour- Standard NLP Group http://nlp.stanford.edu/IR-book/html/htmledition/k-nearest-neighbor-1.html
3. Natural Language Processing with Python http://www.nltk.org/book_1ed/
4. Python Librarieshttps://www.python.org/
5. http://burrsettles.com/pub/settles.activelearning.pdf
6. http://jmlr.csail.mit.edu/proceedings/papers/v16/cawley11a/cawley11a.pdf
7. http://wing.comp.nus.edu.sg/~antho/N/N15/N15-1047.pdf
8. https://en.wikipedia.org/wiki/Active_learning
9. http://citeseerx.ist.psu.edu/showciting?cid=1220400
10. http://www.cs.cornell.edu/people/pabo/movie-review-data/