Machine Learning

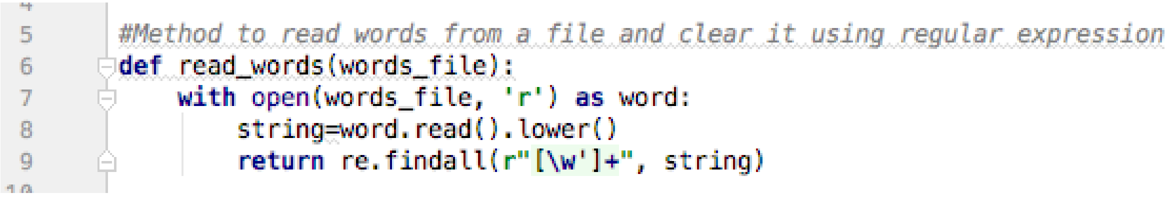
ASSIGNMENT 2

Rushikesh Dixit

1001434559

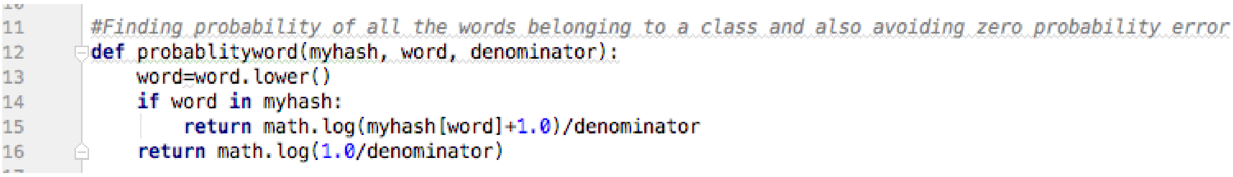
Initial Setup

The Naïve Bayes Classifier is implemented using python on the 20 newsgroups dataset using half as the training data as well as half as test data.



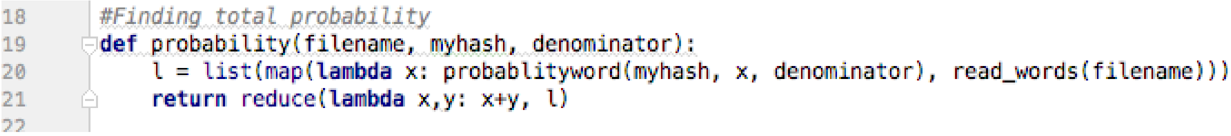
```
#Method to read words from a file and clear it using regular expression
def read_words(words_file):
    with open(words_file, 'r') as word:
        string=word.read().lower()
        return re.findall(r"[\w']+", string)
```

The above is a read words method which opens the file passed as parameter and reads it in lowercase. Also the file is read using the specified regular expression which helps to get rid of unneeded special characters.



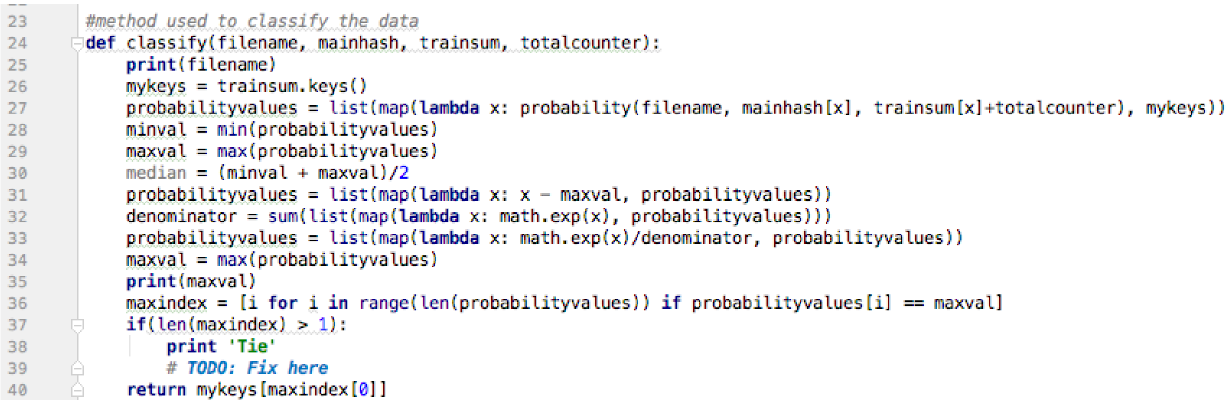
```
#Finding probability of all the words belonging to a class and also avoiding zero probability error
def probablityword(myhash, word, denominator):
    word=word.lower()
    if word in myhash:
        return math.log(myhash[word]+1.0)/denominator
    return math.log(1.0/denominator)
```

Here the above method calculates the probability of each word and also avoids the zero probability error by adding 1 to numerator and adding the total count of words into the denominator.



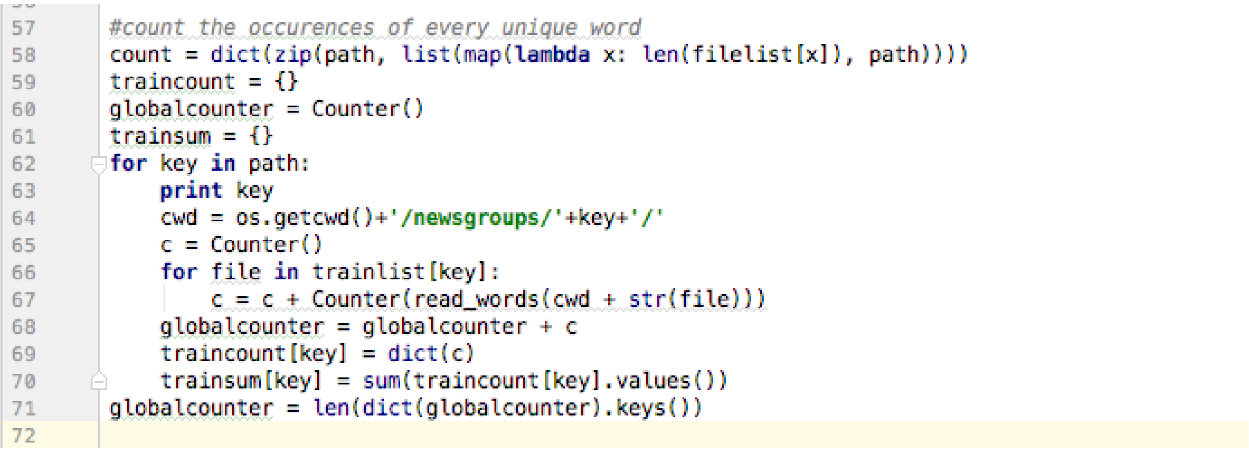
```
#Finding total probability
def probability(filename, myhash, denominator):
    l = list(map(lambda x: probablityword(myhash, x, denominator), read_words(filename)))
    return reduce(lambda x,y: x+y, l)
```

Here the total probability is calculated by using the previous two methods: read words and probabilityword.



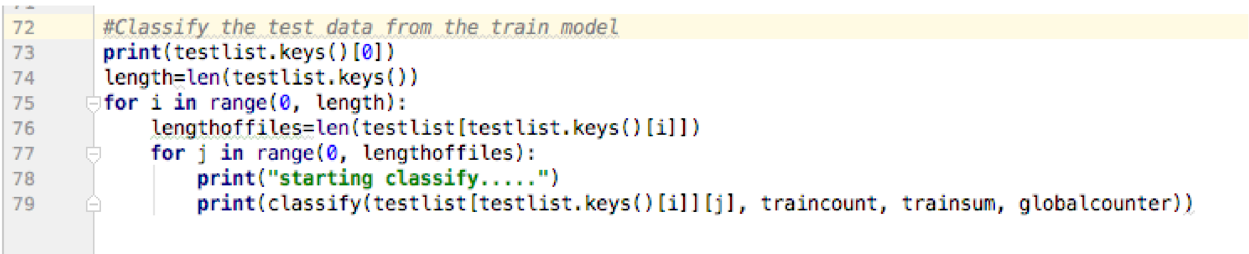
```
#method used to classify the data
def classify(filename, mainhash, trainsum, totalcounter):
    print(filename)
    mykeys = trainsum.keys()
    probabilityvalues = list(map(lambda x: probability(filename, mainhash[x], trainsum[x]+totalcounter), mykeys))
    minval = min(probabilityvalues)
    maxval = max(probabilityvalues)
    median = (minval + maxval)/2
    probabilityvalues = list(map(lambda x: x - maxval, probabilityvalues))
    denominator = sum(list(map(lambda x: math.exp(x), probabilityvalues)))
    probabilityvalues = list(map(lambda x: math.exp(x)/denominator, probabilityvalues))
    maxval = max(probabilityvalues)
    print(maxval)
    maxindex = [i for i in range(len(probabilityvalues)) if probabilityvalues[i] == maxval]
    if(len(maxindex) > 1):
        print 'Tie'
        # TODO: Fix here
    return mykeys[maxindex[0]]
```

The above line of code is used to define the classify method which classifies the test data by calculating max probability values.



```
#count the occurences of every unique word
count = dict(zip(path, list(map(lambda x: len(filelist[x]), path))))
traincount = {}
globalcounter = Counter()
trainsum = {}
for key in path:
    print key
    cwd = os.getcwd()+'/newsgroups/'+key+'/'
    c = Counter()
    for file in trainlist[key]:
        c = c + Counter(read_words(cwd + str(file)))
    globalcounter = globalcounter + c
    traincount[key] = dict(c)
    trainsum[key] = sum(traincount[key].values())
globalcounter = len(dict(globalcounter).keys())
```

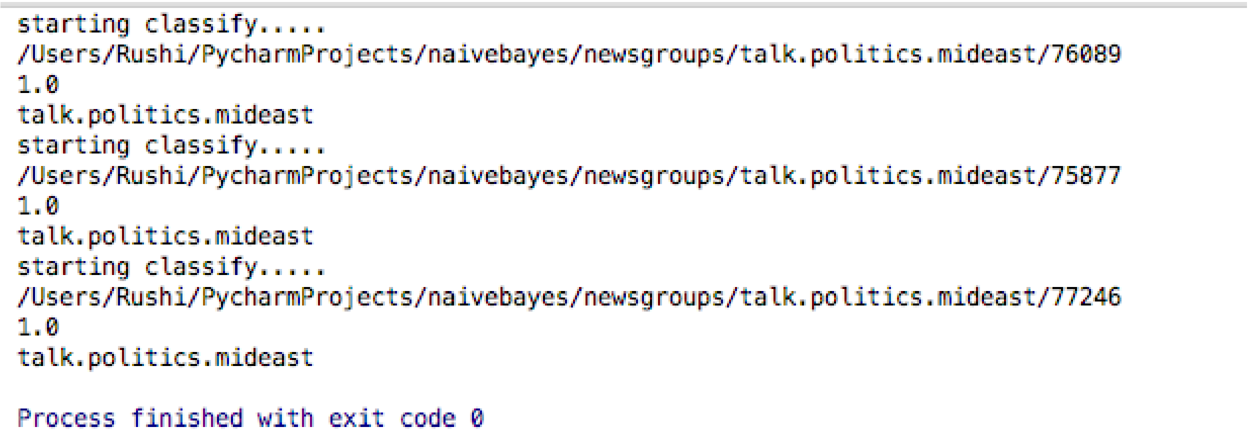
This above snippet is used to count the occurrences of unique word from each document for each class using the counter() method from the Collections package.



```
#Classify the test data from the train model
print(testlist.keys()[0])
length=len(testlist.keys())
for i in range(0, length):
    lengthoffiles=len(testlist[testlist.keys()[i]])
    for j in range(0, lengthoffiles):
        print("starting classify.....")
        print(classify(testlist[testlist.keys()[i]][j], traincount, trainsum, globalcounter))
```

The last part shown above classifies the test data from the testlist which includes half of the documents from each class.

Output:



```
starting classify.....
/Users/Rushi/PycharmProjects/naivebayes/newsgroups/talk.politics.mideast/76089
1.0
talk.politics.mideast
starting classify.....
/Users/Rushi/PycharmProjects/naivebayes/newsgroups/talk.politics.mideast/75877
1.0
talk.politics.mideast
starting classify.....
/Users/Rushi/PycharmProjects/naivebayes/newsgroups/talk.politics.mideast/77246
1.0
talk.politics.mideast

Process finished with exit code 0
```

The above screenshot shows the output of the classifier where it shows the file and its path that is being tested and shows the output of the classifier as talk.politics.mideast.