Homework 8

Sentiment Analysis on IMDB movie reviews

Reference: https://github.com/ikhlaqsidhu/data-x/blob/master/07-tools-webscraping-crawling-nlp-sentiment-sc1t/notebook-nlp-sentiment-analysis-imdb-afo_v1.ipynb (https://github.com/ikhlaqsidhu/data-x/blob/master/07-tools-webscraping-crawling-nlp-sentiment-sc1t/notebook-nlp-sentiment-analysis-imdb-afo_v1.ipynb)

https://github.com/ikhlaqsidhu/data-x/blob/master/07a-tools-nlp-sentiment_add_missing_si/NLP1-slides_v2_afo.pdf (https://github.com/ikhlaqsidhu/data-x/blob/master/07a-tools-nlp-sentiment_add_missing_si/NLP1-slides_v2_afo.pdf)

Name - Miran Tafazzul Hussain Junaidi

SID - 3034487132

As you go through the notebook, you will encounter these main steps in the code:

1. Reading of file labeledTrainData.tsv from data folder in a dataframe `train`.
2. A function review_cleaner(train['review'],lemmatize,stem) which cleans the reviews in the input file.
3. A function train_predict_sentiment(cleaned_reviews, y=train["sentiment"],ngram=1,max_features=1000
4. You will see a model has been trained on unigrams of the reviews without lemmatizing and stemming.
5. Your task is in 5.TODO section.

Run the cells below-

In [1]:

```
# Remove warnings
from __future__ import print_function, division, absolute_import
import warnings
warnings.filterwarnings('ignore')

import matplotlib.pyplot as plt
%matplotlib inline

#make compatible with Python 2 and Python 3
#from __future__ import print_function, division, absolute_import
```

Data set

The labeled training data set consists of 25,000 IMDB movie reviews. There is also an unlabeled test set with 25,000 IMDB movie reviews. The sentiment of the reviews are binary, meaning an IMDB rating < 5 results in a sentiment score of 0, and a rating >=7 have a sentiment score of 1 (no reviews with score 5 or 6 are included in the analysis). No individual movie has more than 30 reviews.

File description

- **labeledTrainData** - The labeled training set. The file is tab-delimited and has a header row followed by 25,000 rows containing an id, sentiment, and text for each review.
- **testData** - The unlabeled test set. 25,000 rows containing an id, and text for each review.

Data columns

- **id** - Unique ID of each review
- **sentiment** - Sentiment of the review; 1 for positive reviews and 0 for negative reviews
- **review** - Text of the review

1. Data set statistics

In [2]:

```


import numpy as np
import pandas as pd
train = pd.read_csv("labeledTrainData.tsv", header=0, \
                    delimiter="\t", quoting=3)
# train.shape should be (25000,3)
```

In [3]:

```
train.head()
```

Out[3]:

	id	sentiment	review
0	"5814_8"	1	"With all this stuff going down at the moment ...
1	"2381_9"	1	"\"The Classic War of the Worlds\" by Timothy ...
2	"7759_3"	0	"The film starts with a manager (Nicholas Bell...
3	"3630_4"	0	"It must be assumed that those who praised thi...
4	"9495_8"	1	"Superbly trashy and wondrously unpretentious ...

In [10]:

```
# import packages

import bs4 as bs
import nltk

#nltk.download('wordnet')
from nltk.tokenize import sent_tokenize # tokenizes sentences
import re

from nltk.stem import PorterStemmer
from nltk.tag import pos_tag
from nltk.corpus import stopwords
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer

eng_stopwords = stopwords.words('english')
```

## 2.Preparing the data set for classification

We'll create a function called `review_cleaner` that reads in a review and:

- Removes HTML tags (using beautifulsoup)
- **Extract emoticons (emotion symbols, aka smileys :D)**
- Removes non-letters (using regular expression)
- Converts all words to lowercase letters and tokenizes them (using .split() method on the review strings, so that every word in the review is an element in a list)
- Removes all the English stopwords from the list of movie review words
- Join the words back into one string seperated by space, append the emoticons to the end

NOTE: Transform the list of stopwords to a set before removing the stopwords. I.e. assign `eng_stopwords = set(stopwords.words("english"))` . Use the set to look up stopwords. This will speed up the computations A LOT (Python is much quicker when searching a set than a list).

In [11]:

```
# 1.
from nltk.corpus import stopwords
from nltk.util import ngrams


ps = PorterStemmer()
wnl = WordNetLemmatizer()


def review_cleaner(reviews,lemmatize=True,stem=False):
    '''
    Clean and preprocess a review.

    1. Remove HTML tags
    2. Use regex to remove all special characters (only keep letters)
    3. Make strings to lower case and tokenize / word split reviews
    4. Remove English stopwords
    5. Rejoin to one string
    '''
    ps = PorterStemmer()
    wnl = WordNetLemmatizer()
        #1. Remove HTML tags

    cleaned_reviews=[]
    for i,review in enumerate(train['review']):
    # print progress
        if( (i+1)%500 == 0 ):
            print("Done with %d reviews" %(i+1))
        review = bs.BeautifulSoup(review).text

        #2. Use regex to find emoticons
        emoticons = re.findall('(?::|;|=)(?:-)?(?:\)|\(|D|P)', review)

        #3. Remove punctuation
        review = re.sub("[^a-zA-Z]", " ",review)

        #4. Tokenize into words (all lower case)
        review = review.lower().split()

        #5. Remove stopwords
        eng_stopwords = set(stopwords.words("english"))

        clean_review=[]
        for word in review:
            if word not in eng_stopwords:
                if lemmatize is True:
                    word=wnl.lemmatize(word)
                elif stem is True:
                    if word == 'oed':
                        continue
                    word=ps.stem(word)
                clean_review.append(word)

        #6. Join the review to one sentence

        review_processed = ' '.join(clean_review+emoticons)
        cleaned_reviews.append(review_processed)
    return(cleaned_reviews)
```

3. Function to train and validate a sentiment analysis model using Random Forest Classifier

In [12]:

```
from sklearn.ensemble import RandomForestClassifier
# # CountVectorizer can actually handle a lot of the preprocessing for us
from sklearn.feature_extraction.text import CountVectorizer
from sklearn import metrics # for confusion matrix, accuracy score etc
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix


np.random.seed(0)


def train_predict_sentiment(cleaned_reviews, y=train["sentiment"],ngram=1,max_featu
    '''This function will:
    1. split data into train and test set.
    2. get n-gram counts from cleaned reviews
    3. train a random forest model using train n-gram counts and y (labels)
    4. test the model on your test split
    5. print accuracy of sentiment prediction on test and training data
    6. print confusion matrix on test data results

    To change n-gram type, set value of ngram argument
    To change the number of features you want the countvectorizer to generate, set t

    print("Creating the bag of words model!\n")
    # CountVectorizer" is scikit-learn's bag of words tool, here we show more keywo
    vectorizer = CountVectorizer(ngram_range=(1, ngram),analyzer = "word",   \
                                 tokenizer = None,    \
                                 preprocessor = None, \
                                 stop_words = None,   \
                                 max_features = max_features)

    X_train, X_test, y_train, y_test = train_test_split(\
    cleaned_reviews, y, random_state=0, test_size=.2)

    # Then we use fit_transform() to fit the model / learn the vocabulary,
    # then transform the data into feature vectors.
    # The input should be a list of strings. .toarraty() converts to a numpy array

    train_bag = vectorizer.fit_transform(X_train).toarray()
    test_bag = vectorizer.transform(X_test).toarray()
#     print('TOP 20 FEATURES ARE: ',(vectorizer.get_feature_names()[:20]))


    print("Training the random forest classifier!\n")
    # Initialize a Random Forest classifier with 75 trees
    forest = RandomForestClassifier(n_estimators = 50)

    # Fit the forest to the training set, using the bag of words as
    # features and the sentiment labels as the target variable
    forest = forest.fit(train_bag, y_train)


    train_predictions = forest.predict(train_bag)
    test_predictions = forest.predict(test_bag)

    train_acc = metrics.accuracy_score(y_train, train_predictions)
    valid_acc = metrics.accuracy_score(y_test, test_predictions)
    print(" The training accuracy is: ", train_acc, "\n", "The validation accuracy i
    print()
    print('CONFUSION MATRIX:')
    print('         Predicted')
    print('          neg pos')
    print(' Actual')
    c=confusion_matrix(y_test, test_predictions)
    print('     neg   ',c[0])
    print('     pos   ',c[1])

    #Extract feature importnace
    print('\nTOP TEN IMPORTANT FEATURES:')
    importances = forest.feature_importances_
    indices = np.argsort(importances)[::-1]
    top_10 = indices[:10]
    print([vectorizer.get_feature_names()[ind] for ind in top_10])
```

4. Train and test Model on the IMDB data

In [13]:

```
#Clean the reviews in the training set 'train' using review_cleaner function define
# Here we use the original reviews without lemmatizing and stemming

original_clean_reviews=review_cleaner(train['review'],lemmatize=False,stem=False)
train_predict_sentiment(cleaned_reviews=original_clean_reviews, y=train["sentiment"]
```

```
Done with 500 reviews
Done with 1000 reviews
Done with 1500 reviews
Done with 2000 reviews
Done with 2500 reviews
Done with 3000 reviews
Done with 3500 reviews
Done with 4000 reviews
Done with 4500 reviews
Done with 5000 reviews
Done with 5500 reviews
Done with 6000 reviews
Done with 6500 reviews
Done with 7000 reviews
Done with 7500 reviews
Done with 8000 reviews
Done with 8500 reviews
Done with 9000 reviews
Done with 9500 reviews
```

5. TODO:

To do this exercise you only need to change argument values in the functions review_cleaner() and train_predict_semtiment(). Go through the functions to understand what they do. Perform the following -

1. For **UNIGRAM setting** ie. when ngram=1 in the function `train_predict_sentiment()` , compare the performance of original cleaned reviews in Sentiment anlysis to -
   A. lemmatized reviews
   B. stemmed reviews
2. For **BIGRAM setting** ie. when ngram=2 in the function `train_predict_sentiment()` , compare the performance of original cleaned reviews in sentiment analysis to:

   A. lemmatized reviews
   B. stemmed reviews
3. For **UNIGRAM setting** ie. ngram=1 and lemmatize = True , compare the performance of Sentiment analysis for these different values of maximum features = [10,100,1000,5000], you can change the value of argument max_features in `train_predict_sentiment()

SUBMISSION: For each question in 5. TODO report your results in a PDF.

Mention the review_cleaner( ) and train_predict_sentiment( ) argument setting that you used in each case. Do not submit any ipython notebook.

Example : For original review with unigram and 5000 max_features, I will report:

```
train_predict_sentiment(cleaned_reviews=original_clean_reviews, y=train["sen
timent"],ngram=1,max_features=5000)

The training accuracy is:  1.0
The validation accuracy is:  0.836
```

Also write a 100-200 word summary of your observations overall.

In [14]:

```
original_clean_reviews=review_cleaner(train['review'],lemmatize=True,stem=False)
train_predict_sentiment(cleaned_reviews=original_clean_reviews, y=train["sentiment"]
```

```
Done with 500 reviews
Done with 1000 reviews
Done with 1500 reviews
Done with 2000 reviews
Done with 2500 reviews
Done with 3000 reviews
Done with 3500 reviews
Done with 4000 reviews
Done with 4500 reviews
Done with 5000 reviews
Done with 5500 reviews
Done with 6000 reviews
Done with 6500 reviews
Done with 7000 reviews
Done with 7500 reviews
Done with 8000 reviews
Done with 8500 reviews
Done with 9000 reviews
Done with 9500 reviews
Done with 10000 reviews
Done with 10500 reviews
Done with 11000 reviews
Done with 11500 reviews
Done with 12000 reviews
Done with 12500 reviews
Done with 13000 reviews
Done with 13500 reviews
Done with 14000 reviews
Done with 14500 reviews
Done with 15000 reviews
Done with 15500 reviews
Done with 16000 reviews
Done with 16500 reviews
Done with 17000 reviews
Done with 17500 reviews
Done with 18000 reviews
Done with 18500 reviews
Done with 19000 reviews
Done with 19500 reviews
Done with 20000 reviews
Done with 20500 reviews
Done with 21000 reviews
Done with 21500 reviews
Done with 22000 reviews
Done with 22500 reviews
Done with 23000 reviews
Done with 23500 reviews
Done with 24000 reviews
Done with 24500 reviews
Done with 25000 reviews
Creating the bag of words model!

Training the random forest classifier!

 The training accuracy is:  0.99995
 The validation accuracy is:  0.8314
```

```
CONFUSION MATRIX:
         Predicted
          neg pos
 Actual
     neg   [2121  427]
     pos   [ 416 2036]

TOP TEN IMPORTANT FEATURES:
['bad', 'worst', 'great', 'waste', 'awful', 'excellent', 'best', 'terr
ible', 'boring', 'nothing']
```

In [15]:

```
original_clean_reviews=review_cleaner(train['review'],lemmatize=False,stem=True)
train_predict_sentiment(cleaned_reviews=original_clean_reviews, y=train["sentiment"]
```

```
Done with 500 reviews
Done with 1000 reviews
Done with 1500 reviews
Done with 2000 reviews
Done with 2500 reviews
Done with 3000 reviews
Done with 3500 reviews
Done with 4000 reviews
Done with 4500 reviews
Done with 5000 reviews
Done with 5500 reviews
Done with 6000 reviews
Done with 6500 reviews
Done with 7000 reviews
Done with 7500 reviews
Done with 8000 reviews
Done with 8500 reviews
Done with 9000 reviews
Done with 9500 reviews
Done with 10000 reviews
Done with 10500 reviews
Done with 11000 reviews
Done with 11500 reviews
Done with 12000 reviews
Done with 12500 reviews
Done with 13000 reviews
Done with 13500 reviews
Done with 14000 reviews
Done with 14500 reviews
Done with 15000 reviews
Done with 15500 reviews
Done with 16000 reviews
Done with 16500 reviews
Done with 17000 reviews
Done with 17500 reviews
Done with 18000 reviews
Done with 18500 reviews
Done with 19000 reviews
Done with 19500 reviews
Done with 20000 reviews
Done with 20500 reviews
Done with 21000 reviews
Done with 21500 reviews
Done with 22000 reviews
Done with 22500 reviews
Done with 23000 reviews
Done with 23500 reviews
Done with 24000 reviews
Done with 24500 reviews
Done with 25000 reviews
Creating the bag of words model!

Training the random forest classifier!

 The training accuracy is:  1.0
 The validation accuracy is:  0.819
```

```
CONFUSION MATRIX:
         Predicted
          neg pos
 Actual
     neg   [2100  448]
     pos   [ 457 1995]

TOP TEN IMPORTANT FEATURES:
['bad', 'worst', 'wast', 'great', 'aw', 'love', 'excel', 'bore', 'terr
ibl', 'best']
```

In [17]:

```
original_clean_reviews=review_cleaner(train['review'],lemmatize=True,stem=False)
train_predict_sentiment(cleaned_reviews=original_clean_reviews, y=train["sentiment"]
```

```
Done with 500 reviews
Done with 1000 reviews
Done with 1500 reviews
Done with 2000 reviews
Done with 2500 reviews
Done with 3000 reviews
Done with 3500 reviews
Done with 4000 reviews
Done with 4500 reviews
Done with 5000 reviews
Done with 5500 reviews
Done with 6000 reviews
Done with 6500 reviews
Done with 7000 reviews
Done with 7500 reviews
Done with 8000 reviews
Done with 8500 reviews
Done with 9000 reviews
Done with 9500 reviews
Done with 10000 reviews
Done with 10500 reviews
Done with 11000 reviews
Done with 11500 reviews
Done with 12000 reviews
Done with 12500 reviews
Done with 13000 reviews
Done with 13500 reviews
Done with 14000 reviews
Done with 14500 reviews
Done with 15000 reviews
Done with 15500 reviews
Done with 16000 reviews
Done with 16500 reviews
Done with 17000 reviews
Done with 17500 reviews
Done with 18000 reviews
Done with 18500 reviews
Done with 19000 reviews
Done with 19500 reviews
Done with 20000 reviews
Done with 20500 reviews
Done with 21000 reviews
Done with 21500 reviews
Done with 22000 reviews
Done with 22500 reviews
Done with 23000 reviews
Done with 23500 reviews
Done with 24000 reviews
Done with 24500 reviews
Done with 25000 reviews
Creating the bag of words model!

Training the random forest classifier!

 The training accuracy is:  1.0
 The validation accuracy is:  0.8238
```

```
CONFUSION MATRIX:
         Predicted
          neg pos
 Actual
     neg   [2112  436]
     pos   [ 445 2007]

TOP TEN IMPORTANT FEATURES:
['bad', 'worst', 'great', 'waste', 'awful', 'terrible', 'boring', 'exc
ellent', 'nothing', 'worse']
```

In [16]:

```
original_clean_reviews=review_cleaner(train['review'],lemmatize=False,stem=True)
train_predict_sentiment(cleaned_reviews=original_clean_reviews, y=train["sentiment"]
```

```
Done with 500 reviews
Done with 1000 reviews
Done with 1500 reviews
Done with 2000 reviews
Done with 2500 reviews
Done with 3000 reviews
Done with 3500 reviews
Done with 4000 reviews
Done with 4500 reviews
Done with 5000 reviews
Done with 5500 reviews
Done with 6000 reviews
Done with 6500 reviews
Done with 7000 reviews
Done with 7500 reviews
Done with 8000 reviews
Done with 8500 reviews
Done with 9000 reviews
Done with 9500 reviews
Done with 10000 reviews
Done with 10500 reviews
Done with 11000 reviews
Done with 11500 reviews
Done with 12000 reviews
Done with 12500 reviews
Done with 13000 reviews
Done with 13500 reviews
Done with 14000 reviews
Done with 14500 reviews
Done with 15000 reviews
Done with 15500 reviews
Done with 16000 reviews
Done with 16500 reviews
Done with 17000 reviews
Done with 17500 reviews
Done with 18000 reviews
Done with 18500 reviews
Done with 19000 reviews
Done with 19500 reviews
Done with 20000 reviews
Done with 20500 reviews
Done with 21000 reviews
Done with 21500 reviews
Done with 22000 reviews
Done with 22500 reviews
Done with 23000 reviews
Done with 23500 reviews
Done with 24000 reviews
Done with 24500 reviews
Done with 25000 reviews
Creating the bag of words model!

Training the random forest classifier!

 The training accuracy is:  1.0
 The validation accuracy is:  0.8238
```

```
CONFUSION MATRIX:
         Predicted
          neg pos
 Actual
     neg   [2111  437]
     pos   [ 444 2008]

TOP TEN IMPORTANT FEATURES:
['worst', 'bad', 'wast', 'great', 'aw', 'bore', 'excel', 'love', 'terr
ibl', 'noth']
```

In [19]:

```
original_clean_reviews=review_cleaner(train['review'],lemmatize=True,stem=False)
train_predict_sentiment(cleaned_reviews=original_clean_reviews, y=train["sentiment"]
```

```
Done with 500 reviews
Done with 1000 reviews
Done with 1500 reviews
Done with 2000 reviews
Done with 2500 reviews
Done with 3000 reviews
Done with 3500 reviews
Done with 4000 reviews
Done with 4500 reviews
Done with 5000 reviews
Done with 5500 reviews
Done with 6000 reviews
Done with 6500 reviews
Done with 7000 reviews
Done with 7500 reviews
Done with 8000 reviews
Done with 8500 reviews
Done with 9000 reviews
Done with 9500 reviews
Done with 10000 reviews
Done with 10500 reviews
Done with 11000 reviews
Done with 11500 reviews
Done with 12000 reviews
Done with 12500 reviews
Done with 13000 reviews
Done with 13500 reviews
Done with 14000 reviews
Done with 14500 reviews
Done with 15000 reviews
Done with 15500 reviews
Done with 16000 reviews
Done with 16500 reviews
Done with 17000 reviews
Done with 17500 reviews
Done with 18000 reviews
Done with 18500 reviews
Done with 19000 reviews
Done with 19500 reviews
Done with 20000 reviews
Done with 20500 reviews
Done with 21000 reviews
Done with 21500 reviews
Done with 22000 reviews
Done with 22500 reviews
Done with 23000 reviews
Done with 23500 reviews
Done with 24000 reviews
Done with 24500 reviews
Done with 25000 reviews
Creating the bag of words model!

Training the random forest classifier!

 The training accuracy is:  0.8714
 The validation accuracy is:  0.5606
```

```
CONFUSION MATRIX:
         Predicted
          neg pos
 Actual
     neg   [1403 1145]
     pos   [1052 1400]

TOP TEN IMPORTANT FEATURES:
['film', 'movie', 'one', 'good', 'character', 'time', 'like', 'get',
'story', 'even']
```

In [20]:

```
original_clean_reviews=review_cleaner(train['review'],lemmatize=True,stem=False)
train_predict_sentiment(cleaned_reviews=original_clean_reviews, y=train["sentiment"]
```

```
Done with 500 reviews
Done with 1000 reviews
Done with 1500 reviews
Done with 2000 reviews
Done with 2500 reviews
Done with 3000 reviews
Done with 3500 reviews
Done with 4000 reviews
Done with 4500 reviews
Done with 5000 reviews
Done with 5500 reviews
Done with 6000 reviews
Done with 6500 reviews
Done with 7000 reviews
Done with 7500 reviews
Done with 8000 reviews
Done with 8500 reviews
Done with 9000 reviews
Done with 9500 reviews
Done with 10000 reviews
Done with 10500 reviews
Done with 11000 reviews
Done with 11500 reviews
Done with 12000 reviews
Done with 12500 reviews
Done with 13000 reviews
Done with 13500 reviews
Done with 14000 reviews
Done with 14500 reviews
Done with 15000 reviews
Done with 15500 reviews
Done with 16000 reviews
Done with 16500 reviews
Done with 17000 reviews
Done with 17500 reviews
Done with 18000 reviews
Done with 18500 reviews
Done with 19000 reviews
Done with 19500 reviews
Done with 20000 reviews
Done with 20500 reviews
Done with 21000 reviews
Done with 21500 reviews
Done with 22000 reviews
Done with 22500 reviews
Done with 23000 reviews
Done with 23500 reviews
Done with 24000 reviews
Done with 24500 reviews
Done with 25000 reviews
Creating the bag of words model!

Training the random forest classifier!

 The training accuracy is:  0.99995
 The validation accuracy is:  0.721
```

```
CONFUSION MATRIX:
         Predicted
          neg pos
 Actual
     neg   [1846  702]
     pos   [ 693 1759]

TOP TEN IMPORTANT FEATURES:
['bad', 'great', 'movie', 'film', 'one', 'best', 'even', 'like', 'lov
e', 'nothing']
```

In [21]:

```
original_clean_reviews=review_cleaner(train['review'],lemmatize=True,stem=False)
train_predict_sentiment(cleaned_reviews=original_clean_reviews, y=train["sentiment"]
```

```
Done with 500 reviews
Done with 1000 reviews
Done with 1500 reviews
Done with 2000 reviews
Done with 2500 reviews
Done with 3000 reviews
Done with 3500 reviews
Done with 4000 reviews
Done with 4500 reviews
Done with 5000 reviews
Done with 5500 reviews
Done with 6000 reviews
Done with 6500 reviews
Done with 7000 reviews
Done with 7500 reviews
Done with 8000 reviews
Done with 8500 reviews
Done with 9000 reviews
Done with 9500 reviews
Done with 10000 reviews
Done with 10500 reviews
Done with 11000 reviews
Done with 11500 reviews
Done with 12000 reviews
Done with 12500 reviews
Done with 13000 reviews
Done with 13500 reviews
Done with 14000 reviews
Done with 14500 reviews
Done with 15000 reviews
Done with 15500 reviews
Done with 16000 reviews
Done with 16500 reviews
Done with 17000 reviews
Done with 17500 reviews
Done with 18000 reviews
Done with 18500 reviews
Done with 19000 reviews
Done with 19500 reviews
Done with 20000 reviews
Done with 20500 reviews
Done with 21000 reviews
Done with 21500 reviews
Done with 22000 reviews
Done with 22500 reviews
Done with 23000 reviews
Done with 23500 reviews
Done with 24000 reviews
Done with 24500 reviews
Done with 25000 reviews
Creating the bag of words model!

Training the random forest classifier!

 The training accuracy is:  1.0
 The validation accuracy is:  0.8182
```

```
CONFUSION MATRIX:
         Predicted
          neg pos
 Actual
     neg   [2103  445]
     pos   [ 464 1988]

TOP TEN IMPORTANT FEATURES:
['bad', 'worst', 'great', 'awful', 'waste', 'excellent', 'terrible',
'wonderful', 'best', 'boring']
```

In [23]:

```
original_clean_reviews=review_cleaner(train['review'],lemmatize=True,stem=False)
train_predict_sentiment(cleaned_reviews=original_clean_reviews, y=train["sentiment"]
```

```
Done with 500 reviews
Done with 1000 reviews
Done with 1500 reviews
Done with 2000 reviews
Done with 2500 reviews
Done with 3000 reviews
Done with 3500 reviews
Done with 4000 reviews
Done with 4500 reviews
Done with 5000 reviews
Done with 5500 reviews
Done with 6000 reviews
Done with 6500 reviews
Done with 7000 reviews
Done with 7500 reviews
Done with 8000 reviews
Done with 8500 reviews
Done with 9000 reviews
Done with 9500 reviews
Done with 10000 reviews
Done with 10500 reviews
Done with 11000 reviews
Done with 11500 reviews
Done with 12000 reviews
Done with 12500 reviews
Done with 13000 reviews
Done with 13500 reviews
Done with 14000 reviews
Done with 14500 reviews
Done with 15000 reviews
Done with 15500 reviews
Done with 16000 reviews
Done with 16500 reviews
Done with 17000 reviews
Done with 17500 reviews
Done with 18000 reviews
Done with 18500 reviews
Done with 19000 reviews
Done with 19500 reviews
Done with 20000 reviews
Done with 20500 reviews
Done with 21000 reviews
Done with 21500 reviews
Done with 22000 reviews
Done with 22500 reviews
Done with 23000 reviews
Done with 23500 reviews
Done with 24000 reviews
Done with 24500 reviews
Done with 25000 reviews
Creating the bag of words model!

Training the random forest classifier!

 The training accuracy is:  1.0
 The validation accuracy is:  0.839
```

```
CONFUSION MATRIX:
         Predicted
          neg pos
 Actual
     neg   [2144  404]
     pos   [ 401 2051]

TOP TEN IMPORTANT FEATURES:
['bad', 'worst', 'great', 'waste', 'awful', 'excellent', 'best', 'noth
ing', 'worse', 'wonderful']
```

Lemmatising and stemming is the process of grouping all the words which typically mean the same and and have different verb form tense or singular/plural. Lemmatising is converting all the influenced words into its root word and Stemming focusses on converting all the influenced words into one word (May or may not be the root word).

A1: When it comes to unigram approach we have noticed that when it comes to train accuracies stemmed reviews have higher accuracy than both the original and lemmatised reviews but that does not nessaraly mean anything since when it comes to validation accuracy lemmatised accuracy is higher than the other 2 and stemmed is less than the original that means lemmatising has actually made the model better and stemming hasn't really done any good.

A2: Now bigram models are actually better and have improved the models but there is no significant difference in either lemmatising or stemming and hence both of them give nearly similar accuracies. However, there is a change in the top 10 important features that means that now different words have more weight in the sentiment.

A3: From changing the max_features from 10 to 5000, we have noticed that there is always a increase in the accuracy with increase in max-features but while comparing it is noticed that there is a significat rise from 10 to 100 and 100 to 1000 but only a 2% increase in the accuracy from 1000 to 5000. and hence i suppose that 1000 will be the