Required Part 0. Sentiment Classification:

Motive:

For part one, I have used “Bag of words as word feature” [A]. I have used 2000 most frequently appearing keywords in the corpus as word features without applying any processing or filter. In this, each feature is contains(keyword) and is true or false as category.

I have compare this with combined features of negation features and varying the representation of the subjectivity lexicon features.

Summary of each features:

A. Baseline feature

In a bag-of-words approach, where we treat each word in the document independent of others and just throw all of them together in the big bag. From one point of view, it looses a lot of information (like how the words are connected), but from another point of view it makes the model simple.

Code:

```
def Baseline_performance_document_features(document,c):
    document_words = set(document)
    features = {}
    for word in Baseline_performance_word_features:
        features['contains(%s)' % word] = (word in document_words)
    return features
Baseline_performance_featuresets = [(Baseline_performance_document_features(d,c), c) for (d,c) in documents]
Features_accuracy_calculation(Baseline_performance_featuresets)
```

B. Combined

Following steps are taken before applying new feature rule

1. Converted text to lower case words to remove upper case and lowercase sensitivity.
2. Remove punctuation and numbers from text as that will deviate our result from correct analysis.
3. Used stop words to remove common words that they do not possess a lot of meaning

```
##############################################################################################
####  Pre-processing the documents  ####
##############################################################################################
def Pre_processing_documents(document):
    # "Pre_processing_documents"
    # "create list of lower case words"
    word_list = re.split('\s+', document.lower())
    # punctuation and numbers to be removed
    punctuation = re.compile(r'[-.?!/\%@,":;()|0-9]')
    stop = stopwords.words('english')
    word_list = [punctuation.sub("", word) for word in word_list]
    final_word_list = []
    for word in word_list:
        if word not in stop:
            final_word_list.append(word)
    stringword = " ".join(final_word_list)
    return stringword
```

New feature rule:

1. **Varied representation of lexicon feature**
   I have created three features that involve counting the positive and negative and neutral word subjectivity words present in each document. Created a feature extraction function that has three extra features ‘positcount’ , ‘negativecount’ and ‘neutral’ with varying importance . I have treated weak positive and weak negative as neutral and giving value 1 out of 2 fraction. For strong positive and strong negative, I am assigning value 2 out of 3 fraction.

   Code:

```
for word in document_words:
    if word in SL:
        strength, posTag, isStemmed, polarity = SL[word]
        if strength == 'weaksubj' and polarity == 'positive':
            weakPos += 1
        if strength == 'strongsubj' and polarity == 'positive':
            strongPos += 1
        if strength == 'weaksubj' and polarity == 'negative':
            weakNeg += 1
        if strength == 'strongsubj' and polarity == 'negative':
            strongNeg += 1
        features['positcount'] = 2/3 * strongPos
        features['negativecount'] = 2/3 * strongNeg
        features['neutral'] = 1/2 * (weakPos  + weakNeg)
```

2. **Negation feature**
   Negation of opinions is an important part of opinion classification. I have tried a simple strategy. We look for negation words "not", "never" and "no" and negation that appears in contractions of the form "don", "", "t". I have taken strategy to go through the document words in order adding the word features, but if the word follows a negation words, change the feature to negated word. Along with this I have taken care of not after some positive word.

Code:

```
#NOT Feature
for word in Negation_word_features:
    features['contains(%s)' % word] = False
    features['contains(NOT%s)' % word] = False

# go through document words in order
for i in range(0, len(document)):
    word = document[i]
    if ((i + 1) < len(document)) and (word in negationwords):
        i += 1
        features['contains(NOT%s)' % document[i]] = (document[i] in processed_words_features)
    elif((i + 2) < len(document)) and (document[i+1] == "not"):
        i += 2
        features['contains(NOT%s)' % document[i]] = (document[i] in processed_words_features)
    else:
        if ((i + 3) < len(document)) and (word.endswith('n') and document[i+1] == "'" and document[i+2] == 't'):
            i += 3
            features['contains(NOT%s)' % document[i]] = (document[i] in processed_words_features)
        else:
            features['contains(%s)' % word] = (word in processed_words_features)
```

Experiments Result:

A. Different size of vocabularies

Output Table for comparison of accuracy by naive Bayesian classifier with same training and test sets size (Training set 95% of data):

size of vocabularies	Bag of words	Combined feature
100	0.55 accuracy	0.705 accuracy
500	0.635 accuracy	0.76 accuracy
1000	0.69 accuracy	0.78 accuracy
2000	0.72 accuracy	0.83 accuracy
2500	0.77 accuracy	0.87 accuracy

B. Obtain precision, recall and F-measure scores

Output Table for comparison of accuracy by naive Bayesian classifier with same training and test sets size (**Training set 95% of data with 2500 size of vocabularies)**:

	Bag of words	Combined feature
Accuracy	0.77	0.87
Precision score	0.825	0.8705
Recall score	0.7522	0.8314
F-measure score	0.7870	0.8505

Conclusion of Result:

Combined feature extraction method has improved accuracy than bag of words method (baseline) method of feature extraction. Along with as the size of vocabularies increase, accuracy of both the feature extraction increases, this is because of larger set of features keys.

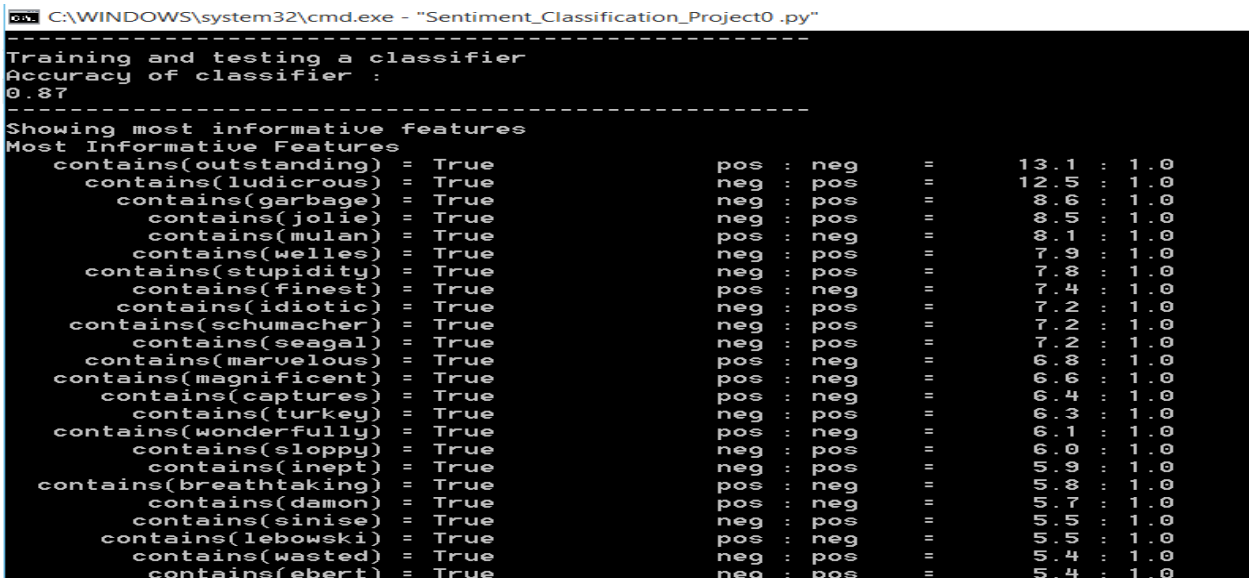
One Sample Output:

```
--------------------------------------------------
Training and testing a classifier
Accuracy of classifier :
0.77
--------------------------------------------------
Showing most informative features
Most Informative Features
        contains(worst) = True              neg : pos    =      4.2 : 1.0
       contains(stupid) = True              neg : pos    =      3.7 : 1.0
       contains(boring) = True              neg : pos    =      3.4 : 1.0
           contains(in) = False             neg : pos    =      2.9 : 1.0
     contains(supposed) = True              neg : pos    =      2.6 : 1.0
      contains(perfect) = True              pos : neg    =      2.3 : 1.0
        contains(oscar) = True              pos : neg    =      2.2 : 1.0
        contains(worse) = True              neg : pos    =      2.2 : 1.0
           contains(it) = False             neg : pos    =      2.2 : 1.0
          contains(bad) = True              neg : pos    =      2.0 : 1.0
       contains(others) = True              pos : neg    =      2.0 : 1.0
 contains(unfortunately) = True             neg : pos    =      2.0 : 1.0
          contains(war) = True              pos : neg    =      1.9 : 1.0
       contains(strong) = True              pos : neg    =      1.9 : 1.0
  contains(performances) = True             pos : neg    =      1.9 : 1.0
        contains(maybe) = True              neg : pos    =      1.9 : 1.0
         contains(none) = True              neg : pos    =      1.9 : 1.0
         contains(true) = True              pos : neg    =      1.9 : 1.0
      contains(attempt) = True              neg : pos    =      1.9 : 1.0
            contains(=) = True              neg : pos    =      1.8 : 1.0
    contains(sometimes) = True              pos : neg    =      1.8 : 1.0
       contains(family) = True              pos : neg    =      1.7 : 1.0
```

```
-------------------------------------------------------
Obtaining precision, recall and F-measure scores

The confusion matrix
     |   n   p |
     |   e   o |
     |   g   s |
----+-------+
neg |<69>18 |
pos | 28<85>|
----+-------+
(row = reference; col = test)

 precision: 0.825242718447
 recall: 0.752212389381
 F-measure: 0.787037037037
```



```
C:\WINDOWS\system32\cmd.exe - "Sentiment_Classification_Project0 .py"
--------------------------------------------------
Training and testing a classifier
Accuracy of classifier :
0.87
--------------------------------------------------
Showing most informative features
Most Informative Features
   contains(outstanding) = True              pos : neg    =     13.1 : 1.0
     contains(ludicrous) = True              neg : pos    =     12.5 : 1.0
       contains(garbage) = True              neg : pos    =      8.6 : 1.0
         contains(jolie) = True              neg : pos    =      8.5 : 1.0
         contains(mulan) = True              pos : neg    =      8.1 : 1.0
        contains(welles) = True              neg : pos    =      7.9 : 1.0
     contains(stupidity) = True              neg : pos    =      7.8 : 1.0
        contains(finest) = True              pos : neg    =      7.4 : 1.0
      contains(idiotic) = True               neg : pos    =      7.2 : 1.0
    contains(schumacher) = True              neg : pos    =      7.2 : 1.0
        contains(seagal) = True              neg : pos    =      7.2 : 1.0
     contains(marvelous) = True              pos : neg    =      6.8 : 1.0
   contains(magnificent) = True              pos : neg    =      6.6 : 1.0
      contains(captures) = True              pos : neg    =      6.4 : 1.0
        contains(turkey) = True              neg : pos    =      6.3 : 1.0
   contains(wonderfully) = True              pos : neg    =      6.1 : 1.0
        contains(sloppy) = True              neg : pos    =      6.0 : 1.0
         contains(inept) = True              neg : pos    =      5.9 : 1.0
  contains(breathtaking) = True              pos : neg    =      5.8 : 1.0
         contains(damon) = True              pos : neg    =      5.7 : 1.0
        contains(sinise) = True              neg : pos    =      5.5 : 1.0
      contains(lebowski) = True              pos : neg    =      5.5 : 1.0
        contains(wasted) = True              neg : pos    =      5.4 : 1.0
         contains(ebert) = True              neg : pos    =      5.4 : 1.0
```

```
--------------------------------------------------------
Obtaining precision, recall and F-measure scores

The confusion matrix
     |    n    p |
     |    e    o |
     |    g    s |
----+----------+
neg |<100>  11  |
pos |   15 <74>|
----+----------+
(row = reference; col = test)

 precision: 0.870588235294
 recall: 0.831460674157
 F-measure: 0.850574712644
```