Kaggle Movie Review Sentiment

Ali Ho & Kendra Osburn | NLP | 6/13/19

HIGHEST ACCURACY ACHIEVED: 94%*

In order to standardize our measurements and conclusions across all experiments, we ran each set of features through three different evaluation measures (**overall accuracy, cross-validation, confusion matrix**) each giving us unique insights into whether or not this new function helped our classification goal.

The **overall accuracy** was simply a Naive Bayes classifier and returned a percentage. This was a good high-level view of our new features.

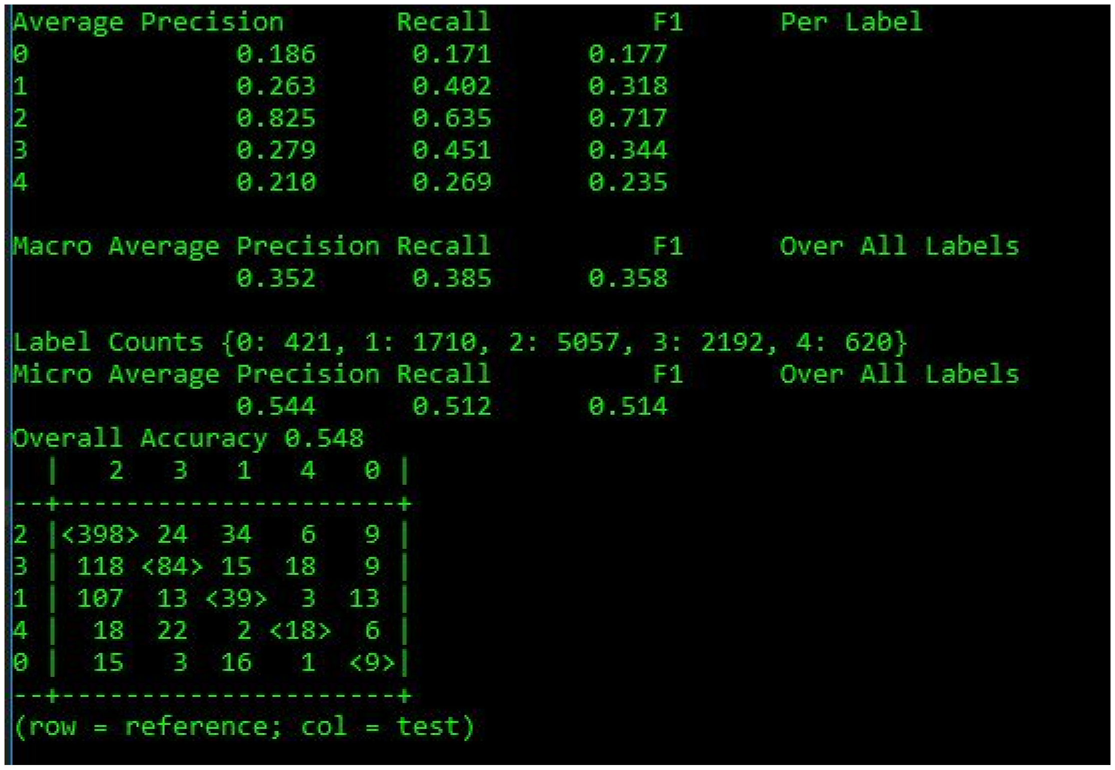
The **cross-validation** took Naive Bayes a step further by breaking the testing and training datasets into "folds." It returned precision, recall, and F1. Recall is calculated by adding up all of the correctly classified movie reviews (the true positives) and dividing it by the sum of the movie reviews that were false positives (for example movie reviews that were predicted to be positive, but were in fact negative) and the true positives. Precision is calculated by adding up all of the correctly classified movie reviews (the true positives) and dividing it by the sum of the movie reviews that were false negatives (movie reviews that are positive, but were not predicted as positive) and the true positives. F1 is the "harmonic mean" of precision and recall. Movie reviews do not have the same level of seriousness with false positives and false negatives as there is with a spam filter. For a spam filter, it is much worse to classify an email as spam, that is in fact not than the other way around.

Lastly, the **confusion matrix** was implemented to ensure enough of each unique grouping made it into the test set, ensuring our data wasn't unintentionally skewed towards any particular sentiment.

EXPERIMENTS: Part One

Testing separate features in separate files

STARTING POINT ACCURACY



```
Average Precision       Recall          F1      Per Label
0              0.186      0.171      0.177
1              0.263      0.402      0.318
2              0.825      0.635      0.717
3              0.279      0.451      0.344
4              0.210      0.269      0.235

Macro Average Precision Recall          F1      Over All Labels
               0.352      0.385      0.358

Label Counts {0: 421, 1: 1710, 2: 5057, 3: 2192, 4: 620}
Micro Average Precision Recall          F1      Over All Labels
               0.544      0.512      0.514
Overall Accuracy 0.548
  |   2   3   1   4   0 |
--+---------------------+
2 |<398> 24  34   6   9 |
3 | 118 <84> 15  18   9 |
1 | 107  13 <39>  3  13 |
4 |  18  22   2 <18>  6 |
0 |  15   3  16   1  <9>|
--+---------------------+
(row = reference; col = test)
```

WHAT WAS IMPLEMENTED

This is the accuracy of the program when we received it. We did change line 201 to obtain a random sample. It was determined that in order to have a true understanding of how the new features are affecting the data a seed needed to be set. This way we will always have the same sample of data that was randomly sampled. This is the best way to compare accuracies.

CODE

```
random.Random(723).shuffle(phrasedata)
```

IN ENGLISH

Random(723) sets a seed in order to replicate the data that was randomly selected every time we run random.shuffle.

CODE

```
train_set, test_set = featuresets[round(.1*int(limit)):], featuresets[:round(.1*int(limit))]
classifier = nltk.NaiveBayesClassifier.train(train_set)
print('Overall Accuracy', nltk.classify.accuracy(classifier, test_set))
```

IN ENGLISH

Creates a test and train dataset. The train set is comprised of 90% of the data and the test set is comprised of the other 10%. It is then run in the Naive Bayes classifier provided by nltk. The output of the classifier is then printed as “Overall Accuracy x%”

CODE

```
goldlist = []
 predictedlist = []
 for (features, label) in test_set:
   goldlist.append(label)
   predictedlist.append(classifier.classify(features))

 cm = nltk.ConfusionMatrix(goldlist, predictedlist)

 print(cm.pretty_format(sort_by_count=True, show_percents=False, truncate = 9))
```

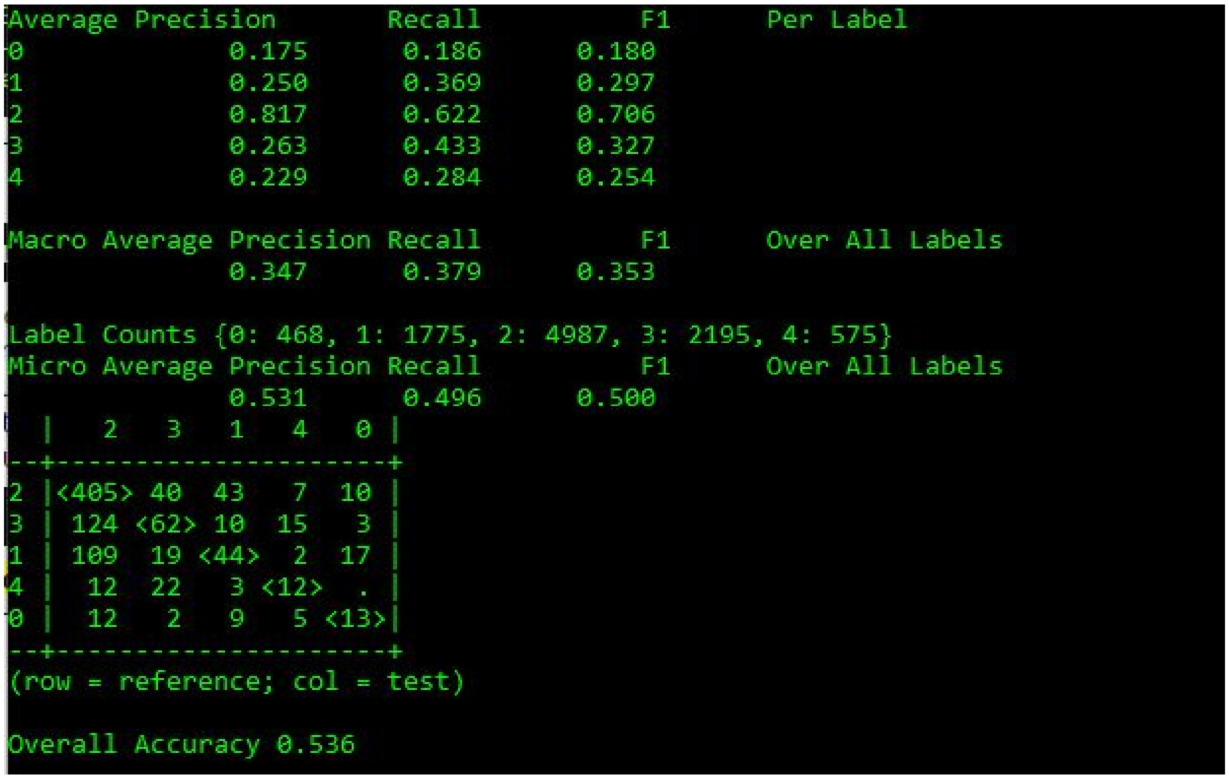
IN ENGLISH

This code creates a confusion matrix. The code has a loop that goes through the test set and compares the actual labels(goldlist) to the predicted labels(predicitedlist). It then uses the ConfusionMatrix function from nltk to create a confusion matrix. The print statement show_percents = False, means that the actual number and not percentage will be shown in the confusion matrix.

REASON

We wanted to have a baseline to compare our results to. Without setting a seed, every time the file is run a different random selection will be generated and the overall accuracy can vary depending on the selection. By setting a random seed, we are able to test how each feature did or did not affect the baseline.

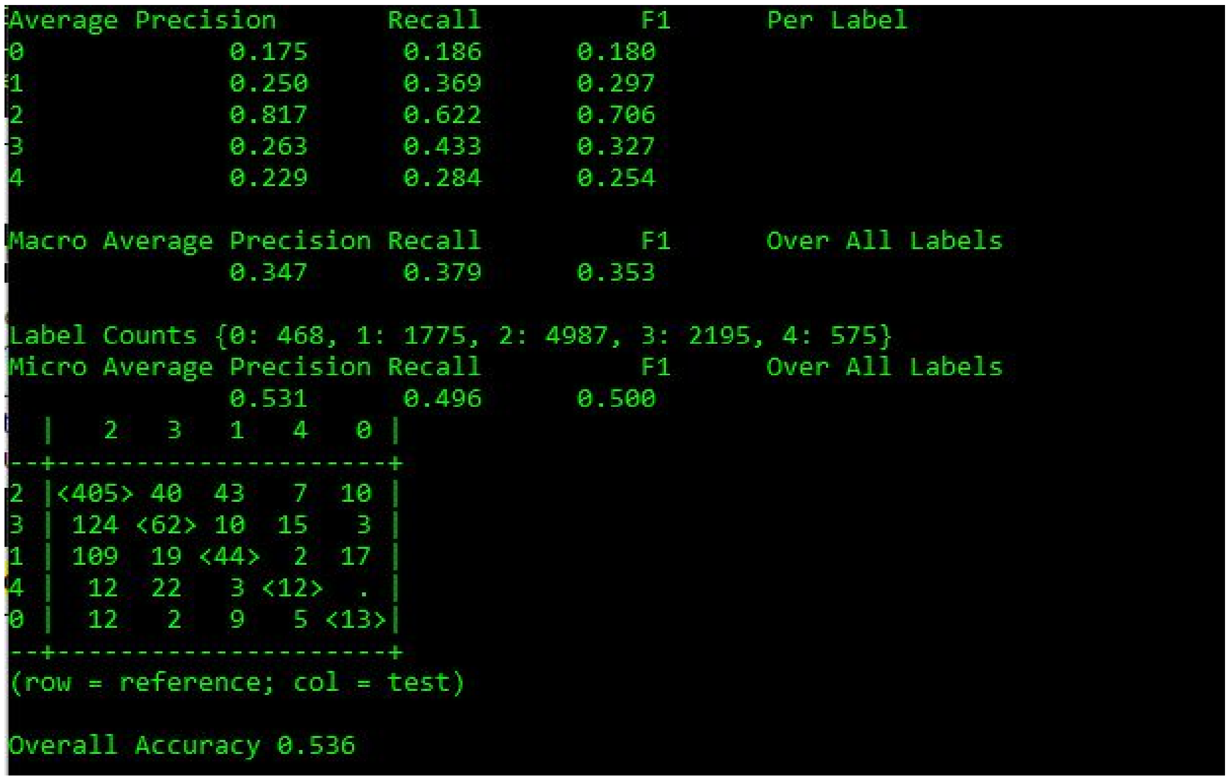
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
0               0.175      0.186      0.180
1               0.250      0.369      0.297
2               0.817      0.622      0.706
3               0.263      0.433      0.327
4               0.229      0.284      0.254

Macro Average Precision Recall          F1      Over All Labels
                0.347      0.379      0.353

Label Counts {0: 468, 1: 1775, 2: 4987, 3: 2195, 4: 575}
Micro Average Precision Recall          F1      Over All Labels
                0.531      0.496      0.500
  |   2   3   1   4   0 |
--+---------------------+
2 |<405> 40  43   7  10 |
3 | 124 <62> 10  15   3 |
1 | 109  19 <44>  2  17 |
4 |  12  22   3 <12>  . |
0 |  12   2   9   5 <13>|
--+---------------------+
(row = reference; col = test)

Overall Accuracy 0.536
```

By creating a confusion matrix, we can see that neutral sentiment are being classified correctly the majority of the time. The reviews that are strongly negative (0) and strongly positive (4) has the lowest success rate for our Naive Bayes classifier. One thing to mention is that this data set is unbalanced, which the majority of our reviews rated as neutral. Therefore, it we should pay attention to the Micro percentages versus the macro. The seed that we chose has a slightly lower overall accuracy then the random sample.

STARTING POINT ACCURACY



```
Average Precision       Recall          F1      Per Label
0              0.175      0.186      0.180
1              0.250      0.369      0.297
2              0.817      0.622      0.706
3              0.263      0.433      0.327
4              0.229      0.284      0.254

Macro Average Precision Recall          F1      Over All Labels
               0.347      0.379      0.353

Label Counts {0: 468, 1: 1775, 2: 4987, 3: 2195, 4: 575}
Micro Average Precision Recall          F1      Over All Labels
               0.531      0.496      0.500
  |   2   3   1   4   0 |
--+---------------------+
2 |<405> 40  43   7  10 |
3 | 124 <62> 10  15   3 |
1 | 109  19 <44>  2  17 |
4 |  12  22   3 <12>  . |
0 |  12   2   9   5 <13>|
--+---------------------+
(row = reference; col = test)

Overall Accuracy 0.536
```

WHAT WAS IMPLEMENTED

We decided to bin the sentiments into 3 different categories: negative, neutral, positive. Reviews that had sentiment scores of:

- 0 or 1 binned as negative
- 2 binned as neutral
- 3 or 4 binned as positive

CODE

```
# each phrase has a list of tokens and the sentiment label (from 0 to 4)
### bin to only 3 categories for better performance
  for phrase in phraselist:
    tokens = nltk.word_tokenize(phrase[0])
    sentiment = int(phrase[1])
    if (sentiment == 2):
      phrasedocs.append((tokens, 'neutral'))
    if ((sentiment == 0) or (sentiment == 1)):
      phrasedocs.append((tokens, 'negative'))
    if ((sentiment == 3) or (sentiment == 4)):
```

```
    phrasedocs.append((tokens, 'positive'))
```

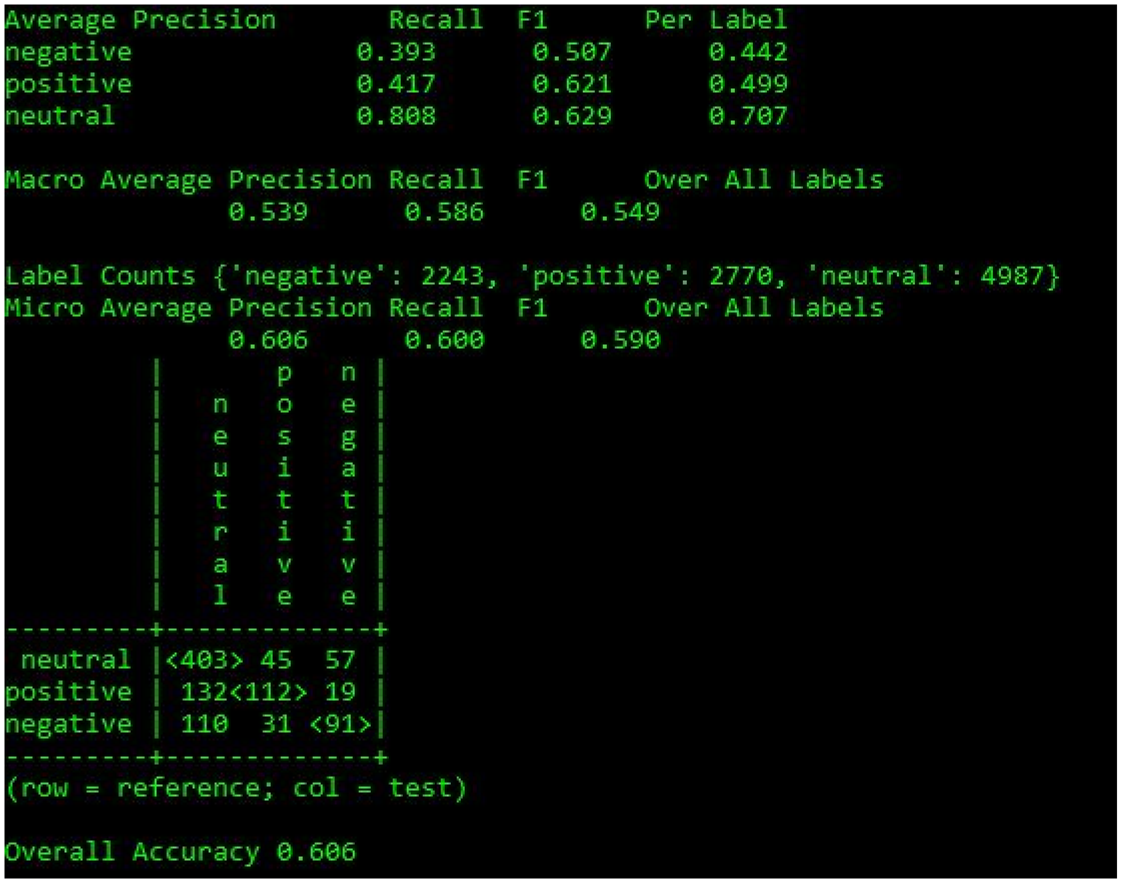
IN ENGLISH

This is a loop that is going through our phraselist and appending it to add either neutral, negative or positive depending on the sentiment value. It then appends the phrasedocs with the tokens for each review phrase and if the review is positive, negative or neutral.

REASON

We decided to bin the data into 3 groups: positive, negative and neutral. The data was originally binned by negative, slightly negative, neutral, slightly positive and positive. However, we are mainly interested if the movie review was negative, neutral or positive, and not on the level of negativity or positivity. Therefore, binning into 3 groups seemed like the best option.

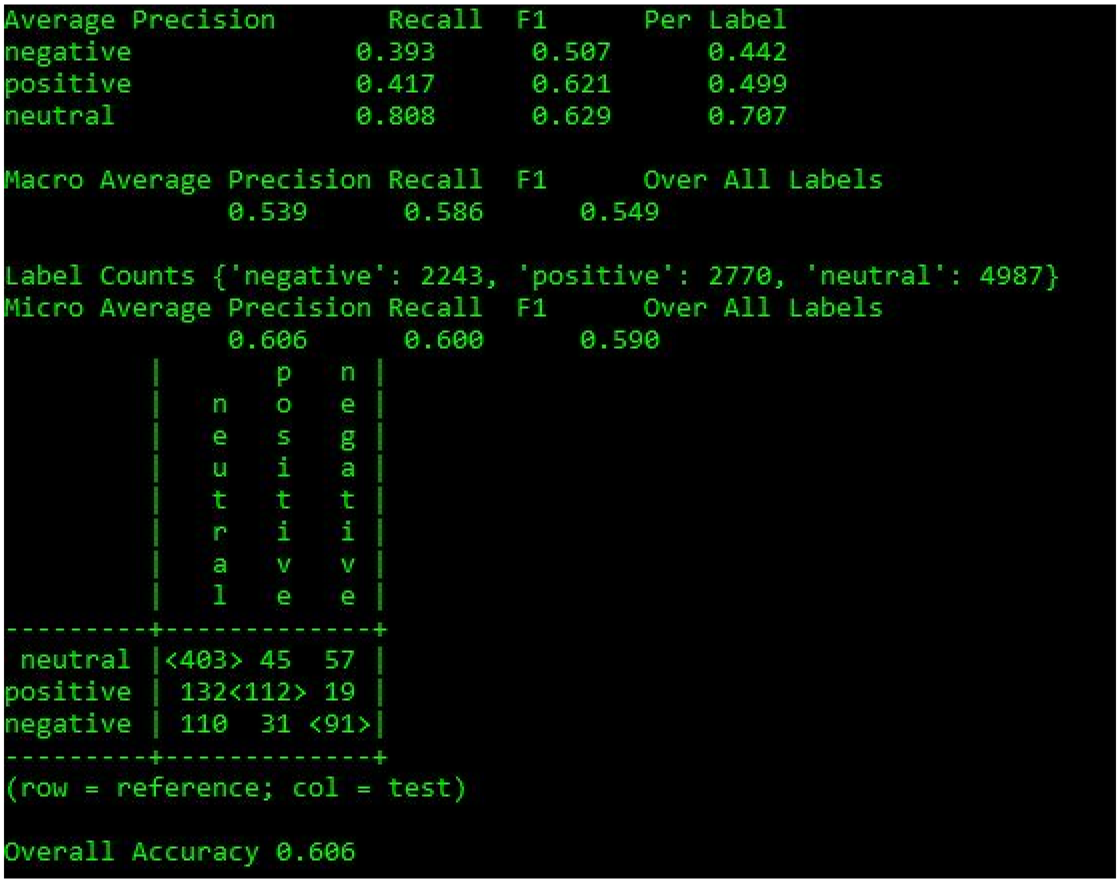
NEW ACCURACY



```
Average Precision       Recall   F1      Per Label
negative              0.393      0.507      0.442
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707

Macro Average Precision Recall   F1      Over All Labels
              0.539      0.586      0.549

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall   F1      Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

Binning into 3 groups increased the overall accuracy from 53.6% to 60.6%. The confusion matrix, shows that the classifier correctly classified 403 neutral phrases out of 505, 112 positive phrases out of 263, and 91 negative reviews out of 232. The classifier classified almost 50% of negative and positive reviews as neutral. This needs to be improved.

STARTING POINT ACCURACY


```
Average Precision       Recall   F1       Per Label
negative              0.393      0.507      0.442
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707

Macro Average Precision Recall   F1       Over All Labels
              0.539      0.586      0.549

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall   F1       Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

WHAT WAS IMPLEMENTED

We decided to remove the neutral bin. All reviews that were classified as neutral were not included in this experiment.

CODE

```
# create list of phrase documents as (list of words, label)
phrasedocs = []
neutraldocs = []
# add all the phrases
# each phrase has a list of tokens and the sentiment label (from 0 to 4)
### bin to only 3 categories for better performance
for phrase in phraselist:
  tokens = nltk.word_tokenize(phrase[0])
#The following code is changing all 0, 1 to "negative", 2 - "neutral", 3 & 4 to "positive"
```

```
#This is essentially binning the phrasedocs into 2 categories: positive and negative
  sentiment = int(phrase[1])
  if (sentiment == 2):
    neutraldocs.append((tokens, 'neutral'))
  if ((sentiment == 0) or (sentiment == 1)):
    phrasedocs.append((tokens, 'negative'))
  if ((sentiment == 3) or (sentiment == 4)):
    phrasedocs.append((tokens, 'positive'))
```

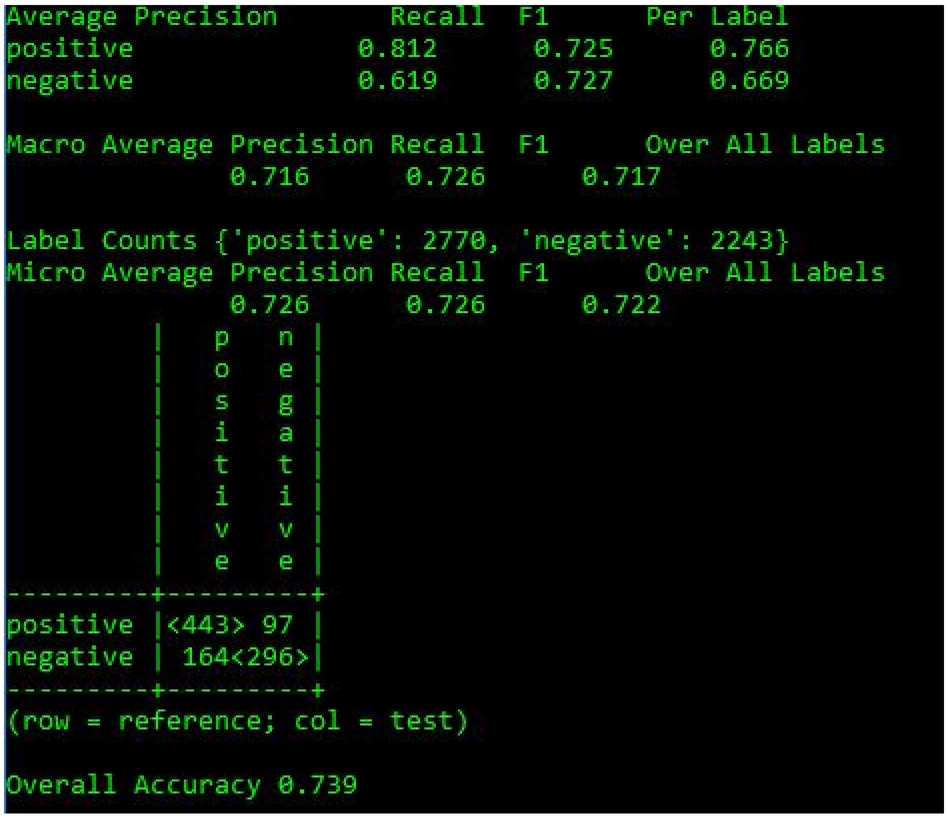
IN ENGLISH

We decided to only append the phrasedocs with the positive and negative reviews. The neutral docs are being added to neutraldocs instead of phrasedocs.

REASON

By looking at the reviews that were broken down to phrases, many phrases were one or two words and the sentiment was neutral because the word was neutral. We are truly only interested in the sentiment for the whole review and not for an individual word. We also are mainly interested in if the review is negative or positive and not neutral. We ultimately want to know if that movie is getting positive reviews and therefore we should go see it.

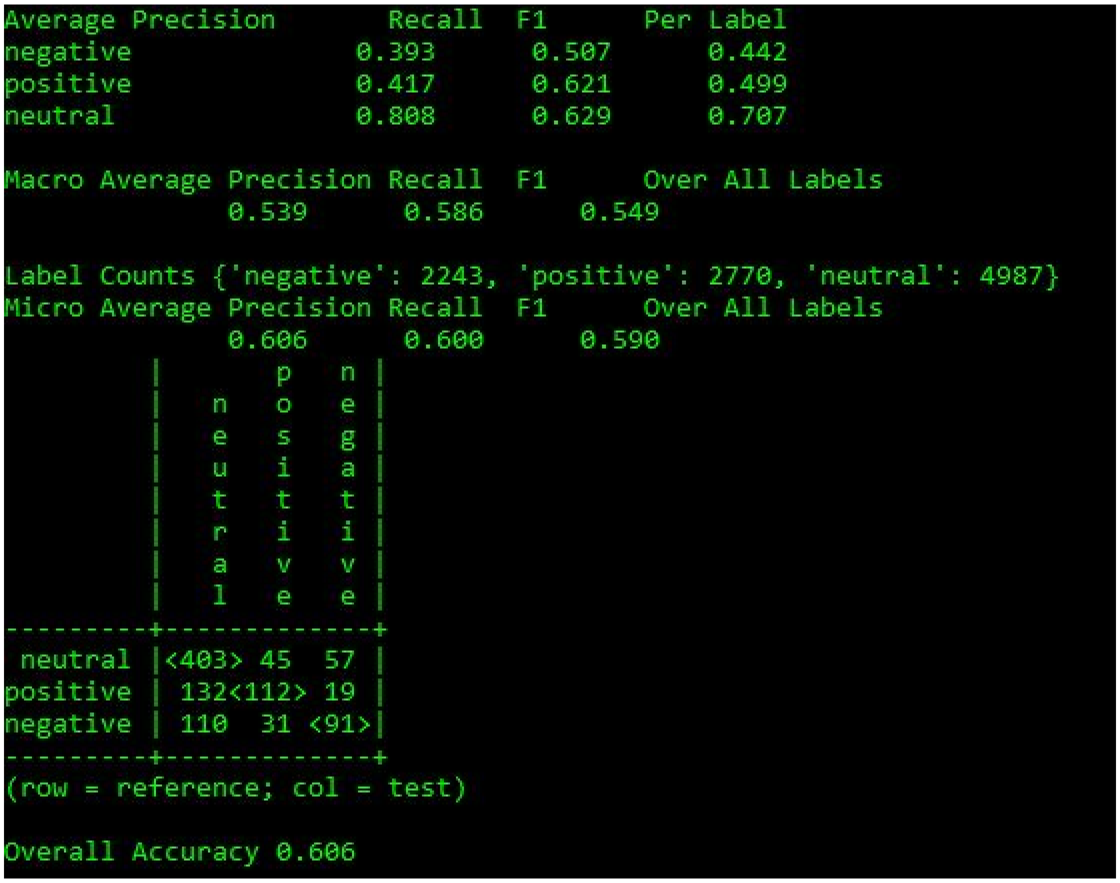
NEW ACCURACY



```
Average Precision       Recall   F1      Per Label
positive              0.812      0.725      0.766
negative              0.619      0.727      0.669

Macro Average Precision Recall   F1      Over All Labels
              0.716      0.726      0.717

Label Counts {'positive': 2770, 'negative': 2243}
Micro Average Precision Recall   F1      Over All Labels
              0.726      0.726      0.722
         |    p   n |
         |    o   e |
         |    s   g |
         |    i   a |
         |    t   t |
         |    i   i |
         |    v   v |
         |    e   e |
---------+---------+
positive |<443> 97 |
negative | 164<296>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.739
```

This greatly increased our accuracy level as we have reduced the noise from the neutral reviews. Our data is also much more balanced and therefore we can look at the macro averages instead of the micro averages. The micro averages are better for unbalanced labels. The confusion matrix, as well as, the average precision per label shows that the classifier had more success correctly classifying positive reviews. We increased the precision, recall and F1 accuracies.

STARTING POINT ACCURACY



```
Average Precision       Recall   F1       Per Label
negative              0.393      0.507      0.442
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707

Macro Average Precision Recall   F1       Over All Labels
              0.539      0.586      0.549

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall   F1       Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

WHAT WAS IMPLEMENTED

Stopwords were removed.

CODE

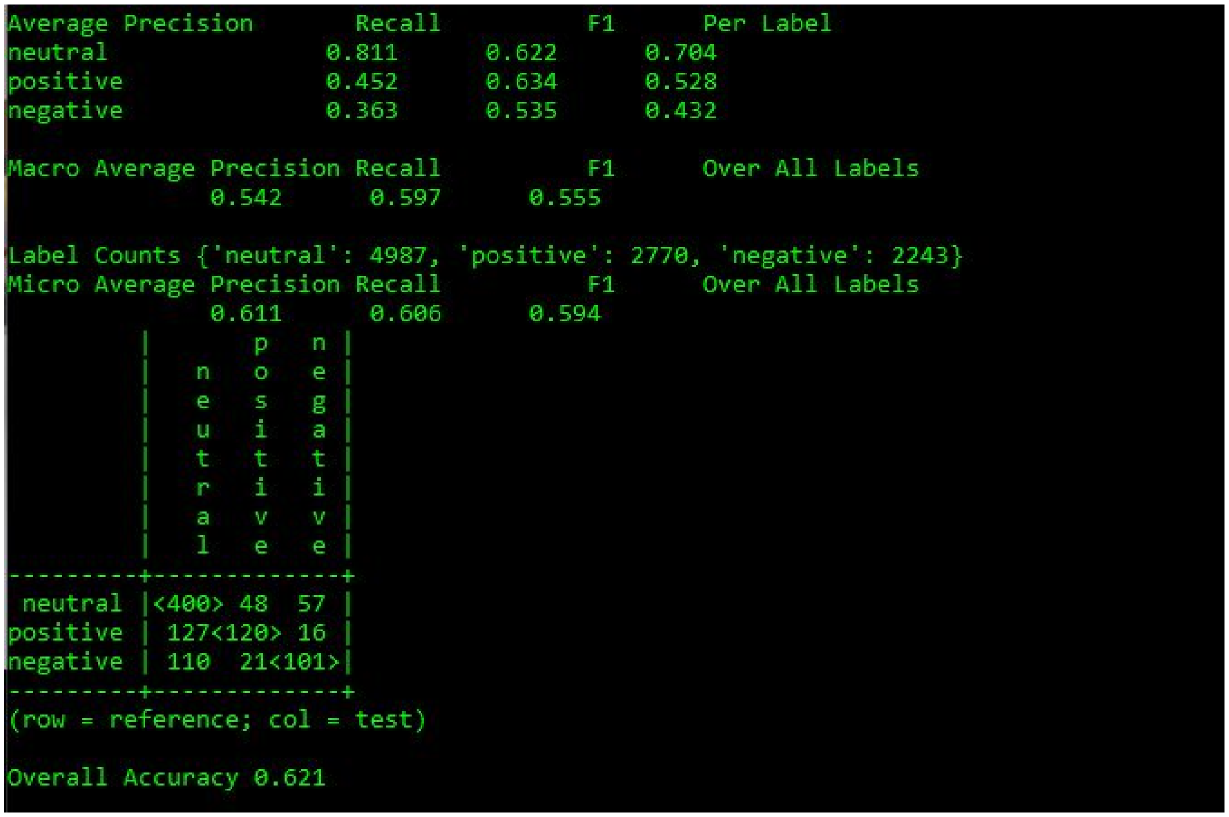
```
from nltk.corpus import stopwords
```

IN ENGLISH
This imports the stopwords list from the nltk.corpus
CODE

`stopwords = nltk.corpus.stopwords.words('english')`
`stopwords.extend([',', '.', '-', 'movie', 'film', '` `', '`', "''", "...", '--'])`

IN ENGLISH
The stopword list from nltk is being saved in an array. We then looked at the top 100 words and decided to add to the stopword list and included some punctuation and the words movie and film.
CODE
all_words_list = [word for (sent, cat) in docs for word in sent **if word not in stopwords**]
IN ENGLISH
This creates a list of all of the words in the reviews that are not in stopwords.
REASON
We believe that stopwords add noise to the movie reviews. We do not feel like the inclusion of stopwords will positively influence our classifier. For this reason, we decided to remove stopwords and see if our intuition is correct.

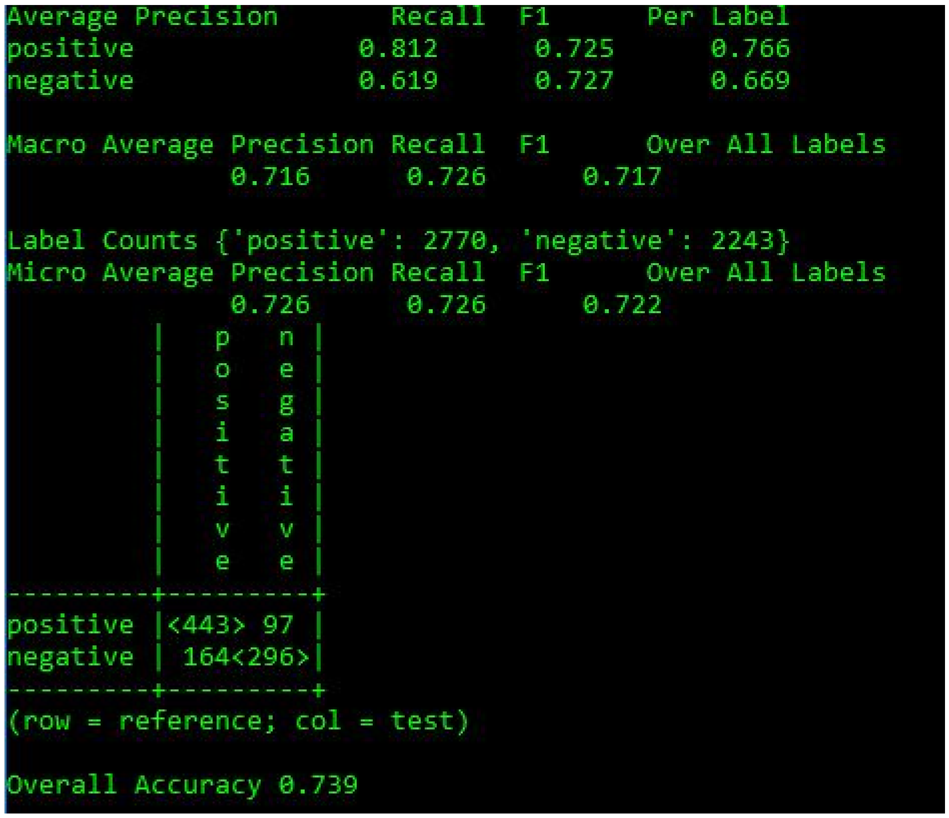
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
neutral               0.811      0.622      0.704
positive              0.452      0.634      0.528
negative              0.363      0.535      0.432

Macro Average Precision Recall          F1      Over All Labels
              0.542      0.597      0.555

Label Counts {'neutral': 4987, 'positive': 2770, 'negative': 2243}
Micro Average Precision Recall          F1      Over All Labels
              0.611      0.606      0.594
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<400> 48  57 |
positive | 127<120> 16 |
negative | 110  21<101>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.621
```

The overall accuracy increased from 60.6% to 62.1%. Removing stopwords appears to have helped our classification task. By removing stopwords, we were able to correctly classify an additional 8 positive reviews, and 10 negative reviews. This increased our micro average precision to 61.1%, recall to 60.6% and F1 to 59.4%. The F1 average scores per label increased for positive and negative, but decreased for neutral. This is acceptable, because ultimately we want a classifier that is able to correctly classify all labels, and not just neutral. We are willing to lose a little bit of accuracy for neutral, but gain precision and recall for positive and negative, which ultimately will increase our F1.

STARTING POINT ACCURACY - NO NEUTRAL

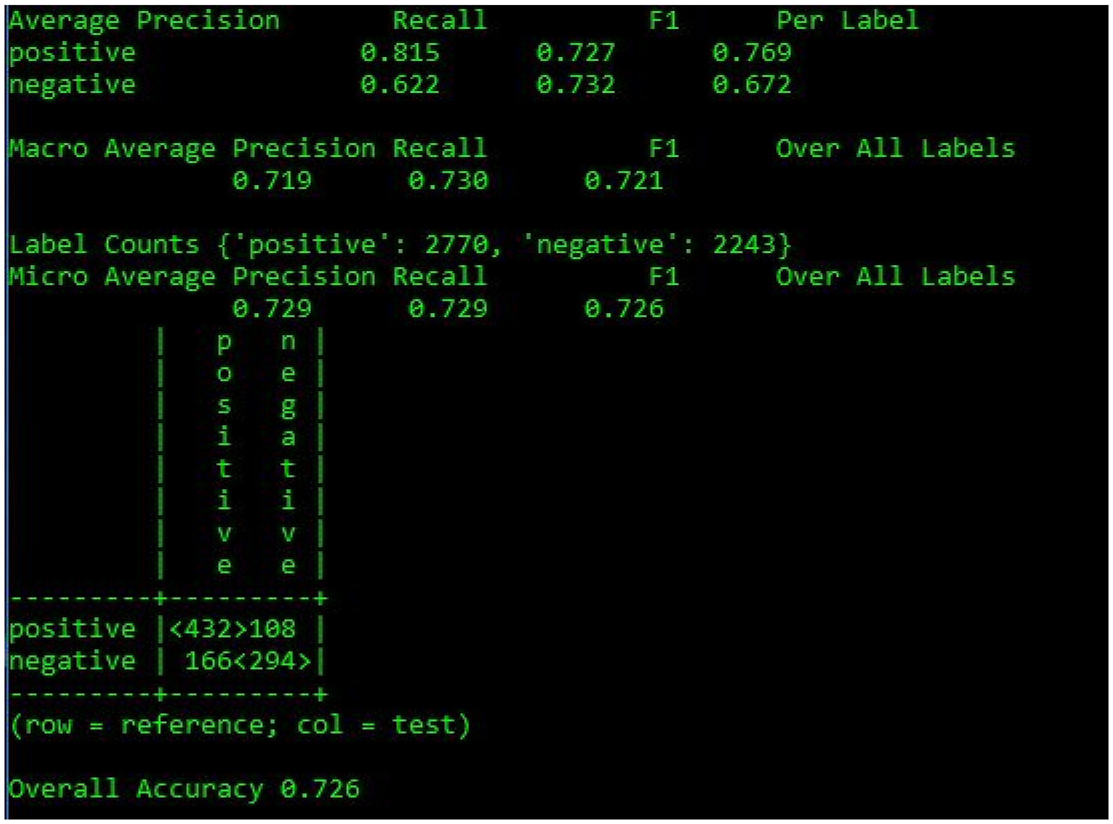


```
Average Precision       Recall  F1      Per Label
positive             0.812      0.725      0.766
negative             0.619      0.727      0.669

Macro Average Precision Recall  F1      Over All Labels
              0.716      0.726      0.717

Label Counts {'positive': 2770, 'negative': 2243}
Micro Average Precision Recall  F1      Over All Labels
              0.726      0.726      0.722
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<443> 97 |
negative | 164<296>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.739
```

WHAT WAS IMPLEMENTED

The stopwords were removed using the steps mentioned above.

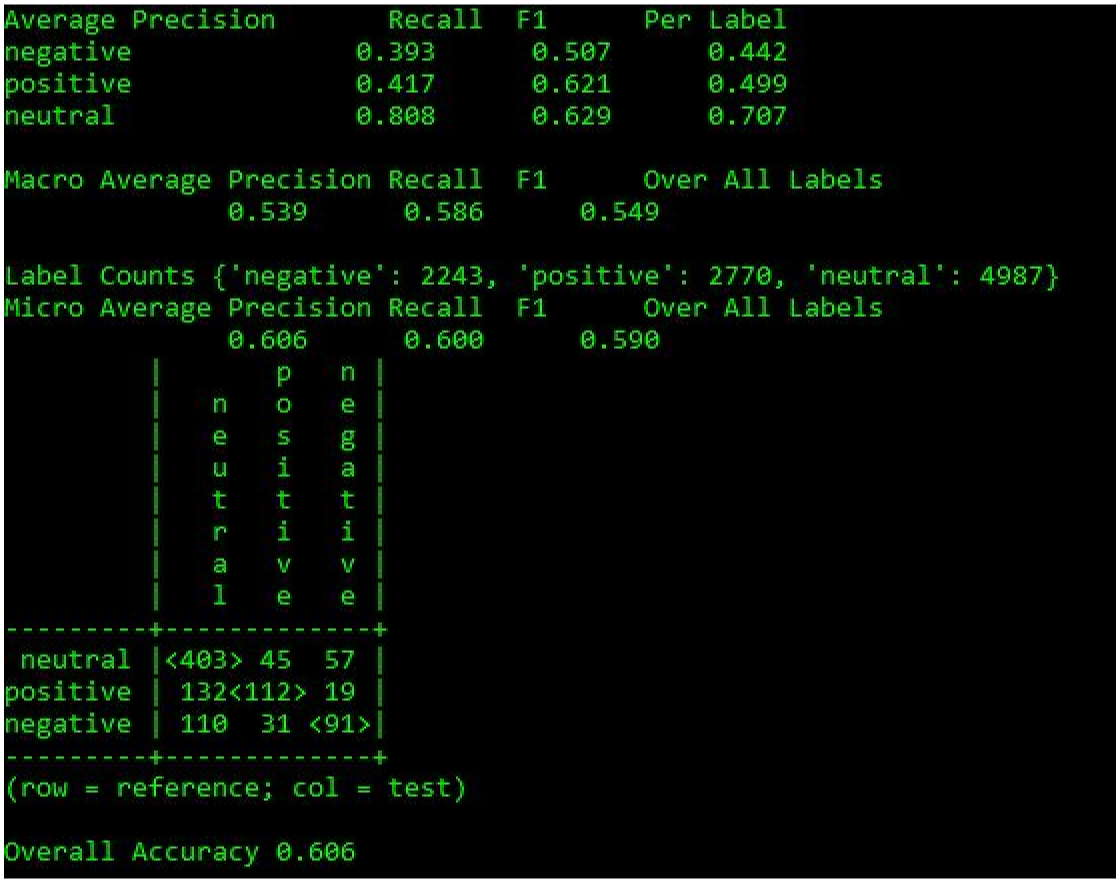
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
positive              0.815      0.727      0.769
negative              0.622      0.732      0.672

Macro Average Precision Recall          F1      Over All Labels
              0.719      0.730      0.721

Label Counts {'positive': 2770, 'negative': 2243}
Micro Average Precision Recall          F1      Over All Labels
              0.729      0.729      0.726
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<432>108 |
negative | 166<294>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.726
```

Interestingly, removing the stopwords for the dataset without neutral labels decreased the overall accuracy. The average precision, increased for both positive and negative movie reviews. Recall also increased for negative reviews. However, the F1 decreased for both negative and positive reviews. Stopwords appear to be important when classifying for only positive and negative reviews. When stopwords were included we were able to correctly classify 11 more positive reviews and 2 more negative reviews. We had an increase of 11 false negatives with stopwords included and 2 more false negatives.

STARTING POINT ACCURACY



```
Average Precision       Recall  F1      Per Label
negative              0.393      0.507      0.442
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707

Macro Average Precision Recall  F1      Over All Labels
              0.539      0.586      0.549

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall  F1      Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

WHAT WAS IMPLEMENTED

Negation was included. This attempt does not remove stopwords and the base file is the binned python file. No other changes were made.

CODE

```
def Not_features(document, word_features, negationwords):
  features = {}
  for word in word_features:
        features['V_{}'.format(word)] = False
        features['V_NOT{}'.format(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) or
(word.endswith("n't"))):
        i += 1
        features['V_NOT{}'.format(document[i])] = (document[i] in word_features)
        else:
        features['V_{}'.format(word)] = (word in word_features)
  return features
```

IN ENGLISH

This defines a negation function that will go through every word in the word features and negate the word that follows a negation word or “n’t”.

CODE

```
negationwords = ['no', 'not', 'never', 'none', 'nowhere', 'nothing', 'noone', 'rather', 'hardly', 'scarcely', 'rarely', 'seldom', 'neither', 'nor']
```

IN ENGLISH

Creates a list of the negation words that we listed

CODE

```
NOT_featuresets = [(Not_features(d, word_features, negationwords), c) for (d, c) in docs]
```

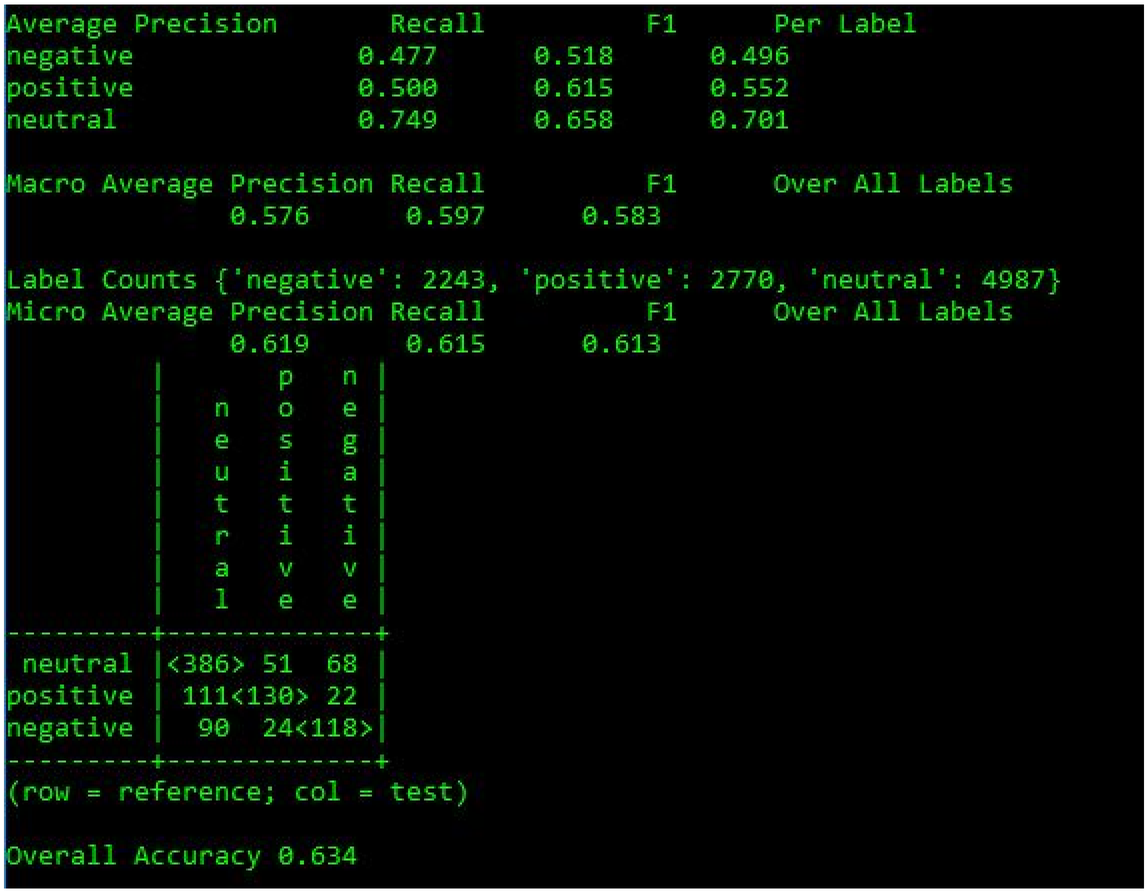
IN ENGLISH

Calls the Not_features function that was defined above. The NOT_featuresets is generating an array. Each item in the array has both an object of features and the sentiment. The object of features contains every word in word_features so V_word : TRUE, V_NOTword : FALSE. It states whether or not that word follows a negation word and if the word is in the phrase.

REASON

We felt that negative reviews would have more negation in them than positive or neutral reviews. If this is the case, we expect to see a higher precision and recall for negative reviews and possibly a lower precision for neutral reviews.

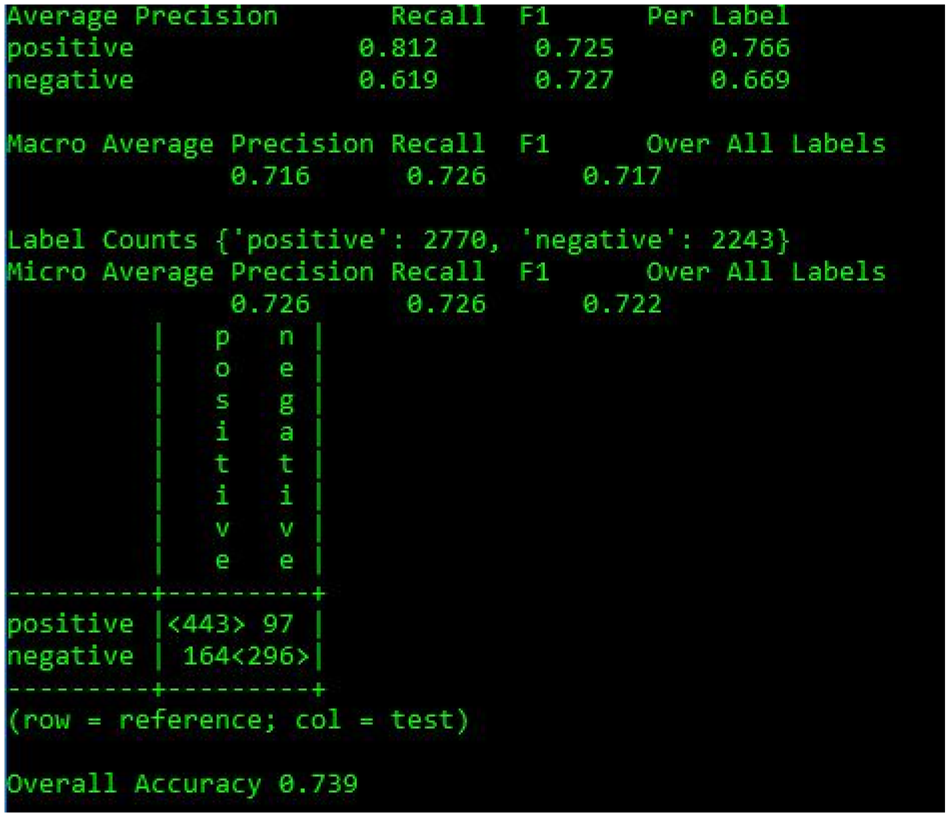
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
negative             0.477      0.518      0.496
positive             0.500      0.615      0.552
neutral              0.749      0.658      0.701

Macro Average Precision Recall          F1      Over All Labels
              0.576      0.597      0.583

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall          F1      Over All Labels
              0.619      0.615      0.613
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<386> 51  68 |
positive | 111<130> 22 |
negative |  90  24<118>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.634
```

Negation improved our macro averages for precision, recall and F1. It did decrease the neutral average precision, but increased the recall. The increase in recall shows that the classifier is not assigning everything to neutral, but in fact is being more selective. Precision for neutral decreased because we had previously predicted 403 neutral reviews correctly, but now are only correctly classifying 386 neutral reviews. This is acceptable because the initial version classified 645 reviews as neutral and this version with negation only classified 587 reviews as neutral. Which means that it classified more reviews than before as either positive or negative. We were able to correctly classify 18 more positive reviews with negation and 27 more negative reviews. Negation was very beneficial in classifying positive and negative reviews.

STARTING POINT ACCURACY - NO NEUTRAL

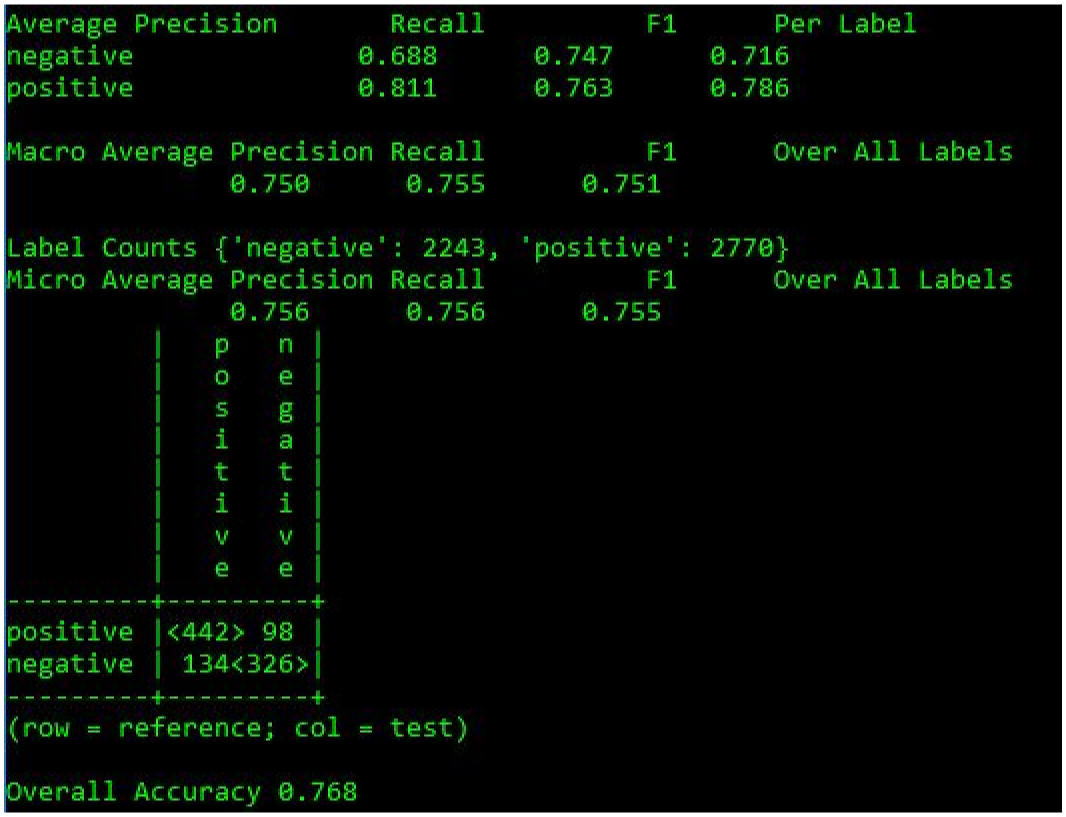


```
Average Precision       Recall  F1      Per Label
positive              0.812      0.725      0.766
negative              0.619      0.727      0.669

Macro Average Precision Recall  F1      Over All Labels
              0.716      0.726      0.717

Label Counts {'positive': 2770, 'negative': 2243}
Micro Average Precision Recall  F1      Over All Labels
              0.726      0.726      0.722
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<443> 97 |
negative | 164<296>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.739
```

WHAT WAS IMPLEMENTED

Negation was included. This attempt does not remove stopwords and the base file is the binned python file. No other changes were made. The steps were the same as the steps listed above for negation.

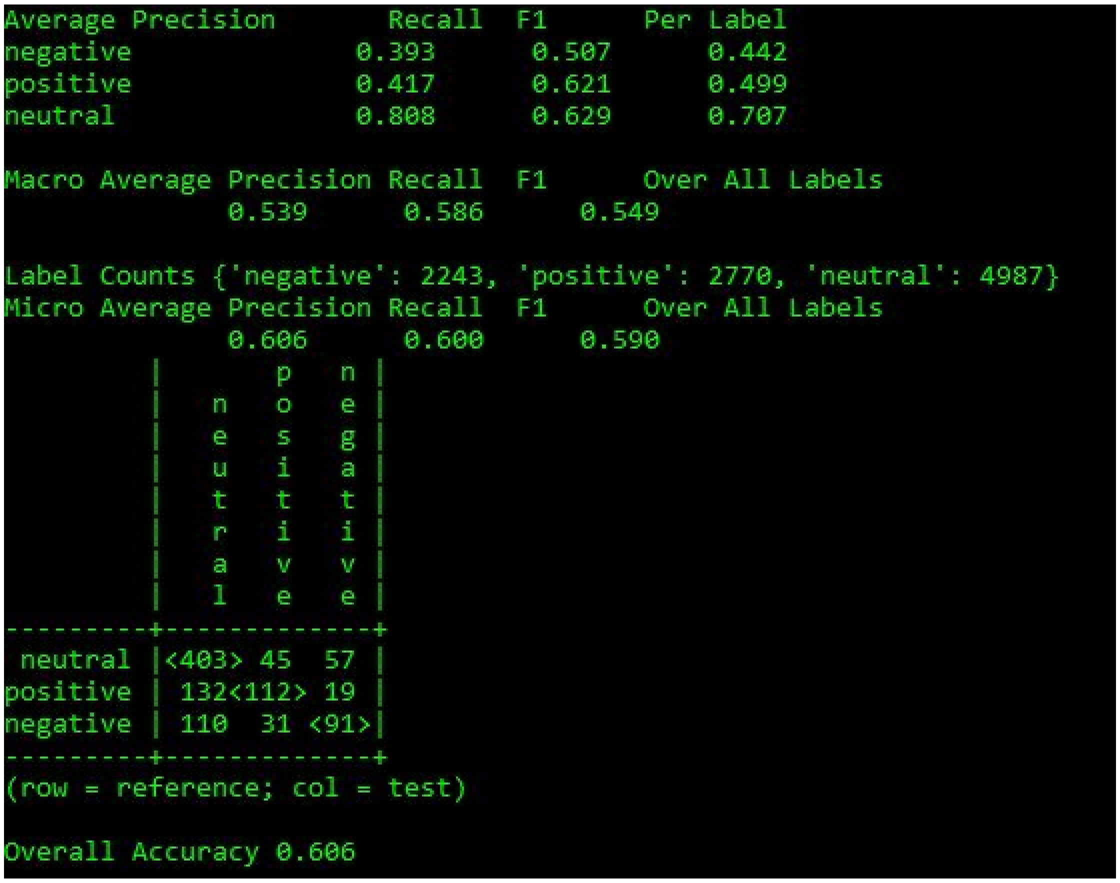
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
negative              0.688      0.747      0.716
positive              0.811      0.763      0.786

Macro Average Precision Recall          F1      Over All Labels
              0.750      0.755      0.751

Label Counts {'negative': 2243, 'positive': 2770}
Micro Average Precision Recall          F1      Over All Labels
              0.756      0.756      0.755
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<442> 98 |
negative | 134<326>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.768
```

Negation yet again proved fruitful for classifying positive and negative movie reviews. Prior to negation the classifier was classifying the majority of movie reviews as positive. The initial version classified 607 of the 1,000 movie reviews as positive, where in actuality there are only 540 positive movie reviews. With negation this was slightly corrected and therefore precision, recall and F1 increased for both positive and negative movie reviews. The macro averages also all increased. We were successfully able to classify 30 more negative movie reviews and only lost 1 correctly classified positive movie review. This was a successful attempt. The combination of bigrams to negation did not change any of the accuracies or predictions when compared to the negation file.

STARTING POINT ACCURACY



```
Average Precision       Recall  F1      Per Label
negative              0.393      0.507      0.442
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707

Macro Average Precision Recall  F1      Over All Labels
              0.539      0.586      0.549

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall  F1      Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

WHAT WAS IMPLEMENTED

Bigrams were implemented to the python file with binned data. It is important to note that when creating bigrams, you cannot remove stopwords as then the bigrams would not be accurate.

CODE

from nltk.collocations import *
bigram_measures = nltk.collocations.BigramAssocMeasures()

IN ENGLISH

This imports the nltk.collocations.BigramAssocMeasures from the nltk.collocations and saves it in bigram_measures.

CODE

def bigram_document_features(document, word_features, bigram_features):

```
  document_words = set(document)
  document_bigrams = nltk.bigrams(document)
  features = {}
  for word in word_features:
        features['V_{}'.format(word)] = (word in document_words)
  for bigram in bigram_features:
        features['B_{}_{}'.format(bigram[0], bigram[1])] = (bigram in
document_bigrams)
  return features
```

IN ENGLISH

This defines a bigram function that contains both word features and bigram features. There are two loops in this function. The first loop goes through every word in the phrases and creates a sparse matrix with V_word and states True or False depending if the word is in that specific phrase. The second loop creates a sparse matrix of bigrams V_word_word and states true or false depending if that bigram is in the phrase.

CODE

```
  finder = BigramCollocationFinder.from_words(all_words_list)
```

IN ENGLISH

This line goes through all of the words in the all_words_list and creates bigrams and stores them in an array called finder.

CODE

```
bigram_features = finder.nbest(bigram_measures.pmi, 500)
```

IN ENGLISH

This line evaluates the bigrams using the pmi method and returns the top 500 bigrams based on their pmi score.

CODE

```
bigram_featuresets = [(bigram_document_features(d, word_features,
bigram_features), c) for (d, c) in docs]
```

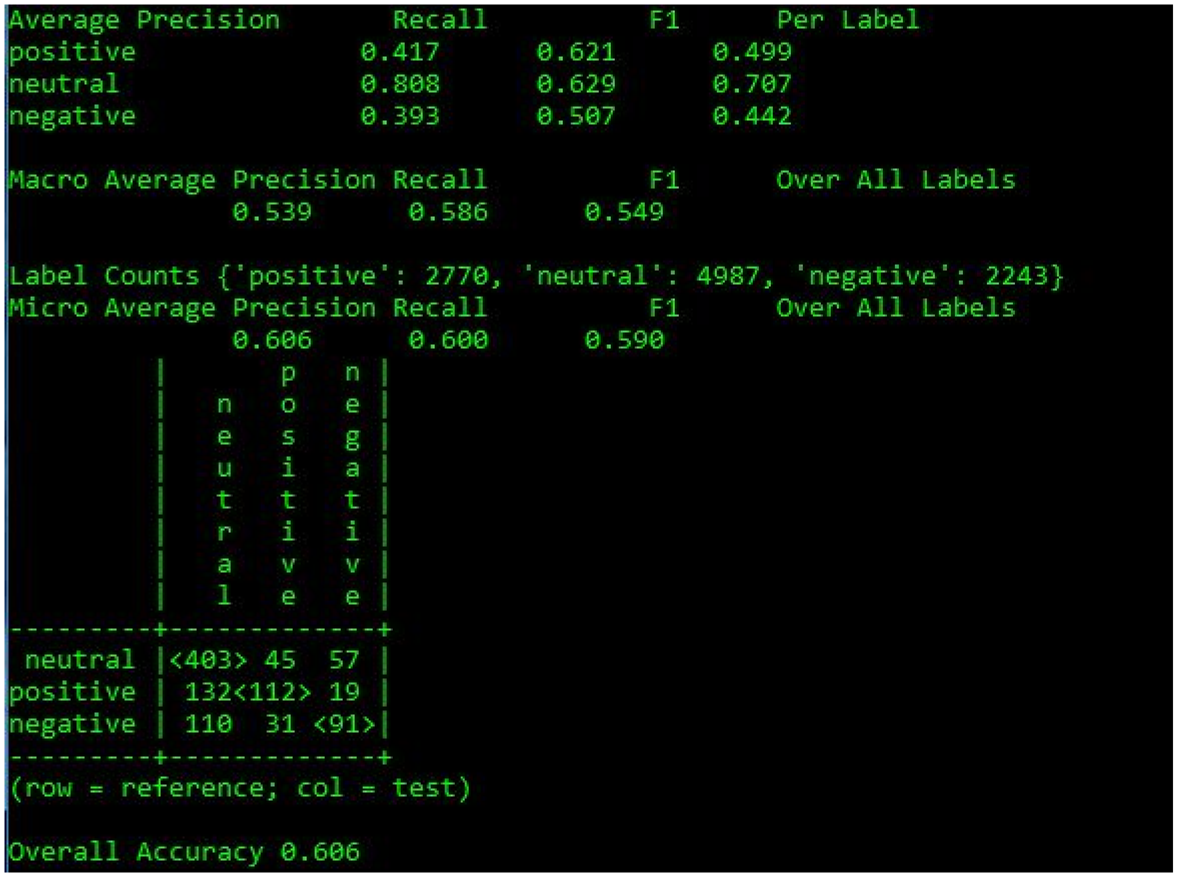
IN ENGLISH

Calls the bigram_document_features function that was defined above. The bigram_document_features is generating an array. Each item in the array has both an object of features and the sentiment. The object of features contains every word in the phrases followed by true or false, depending on if the word is represented in that specific phrase, it also contains the top 500 bigrams with a true or false, depending on if the bigram appears in that phrase followed by the sentiment.

REASONS

Bigrams are an important tool used in sentiment classification. However, we wonder how helpful they will be in this instance, because the phrases are broken down into multiple phrases and sentences are not kept together.

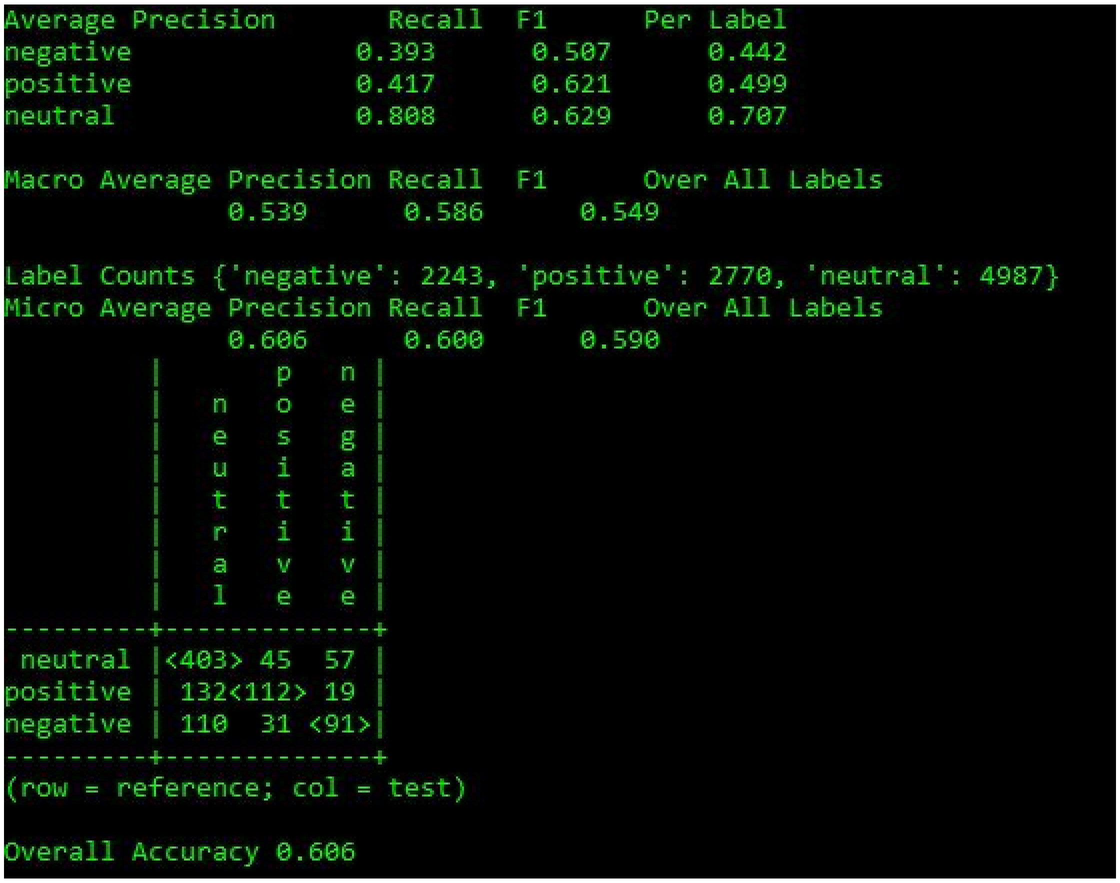
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707
negative              0.393      0.507      0.442

Macro Average Precision Recall          F1      Over All Labels
              0.539      0.586      0.549

Label Counts {'positive': 2770, 'neutral': 4987, 'negative': 2243}
Micro Average Precision Recall          F1      Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

Implementing bigrams with a pmi score, did not change the classification accuracy at all. We will experiment to see if either the bigrams with a raw frequency or chi square scores will prove beneficial.

STARTING POINT ACCURACY



```
Average Precision       Recall  F1      Per Label
negative              0.393      0.507      0.442
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707

Macro Average Precision Recall  F1      Over All Labels
              0.539      0.586      0.549

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall  F1      Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

WHAT WAS IMPLEMENTED

Bigrams were implemented to the python file with binned data. It is important to note that when creating bigrams, you cannot remove stopwords as then the bigrams would not be accurate. This attempt uses the chi_sq measure

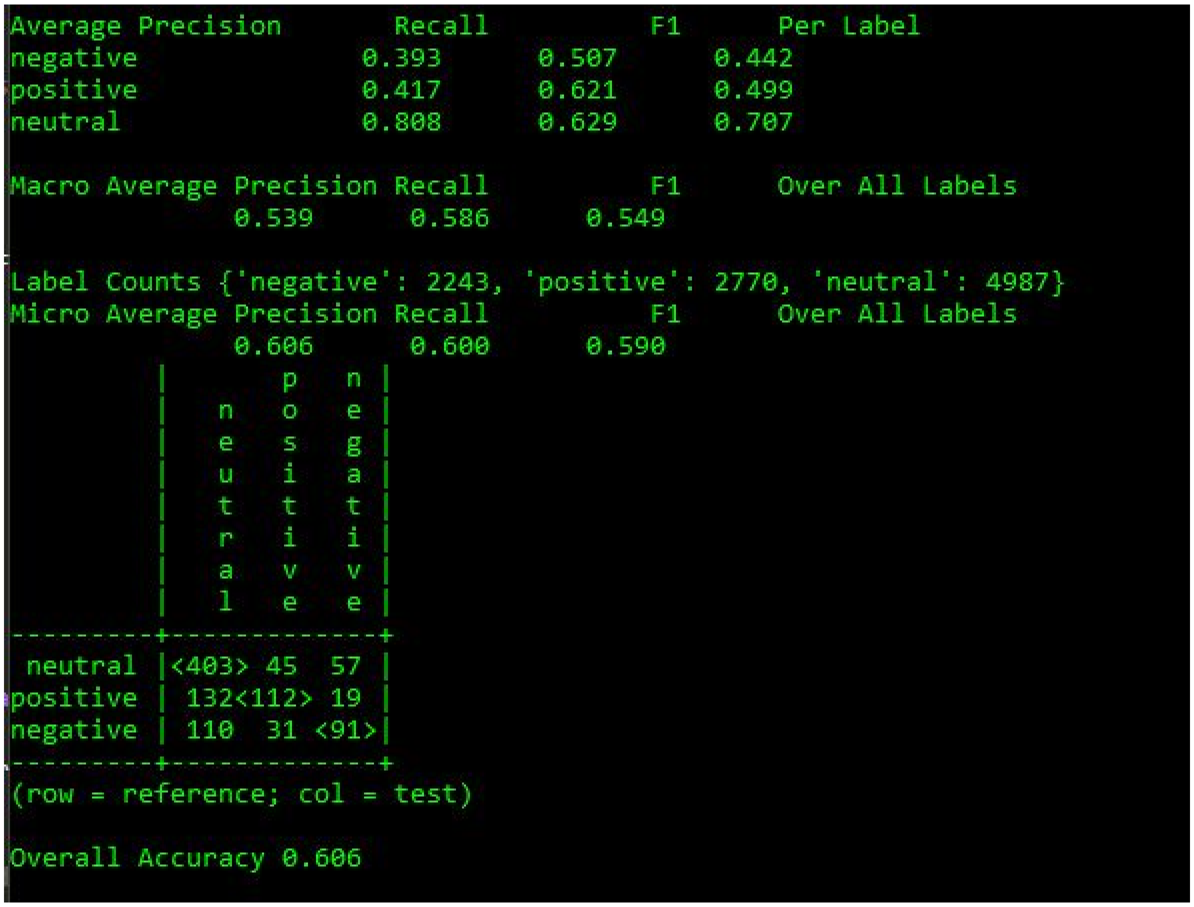
CODE

bigram_features = finder.nbest(bigram_measures.chi_sq, 500)

IN ENGLISH

This line evaluates the bigrams using the pmi method and returns the top 500 bigrams based on their chi square score.

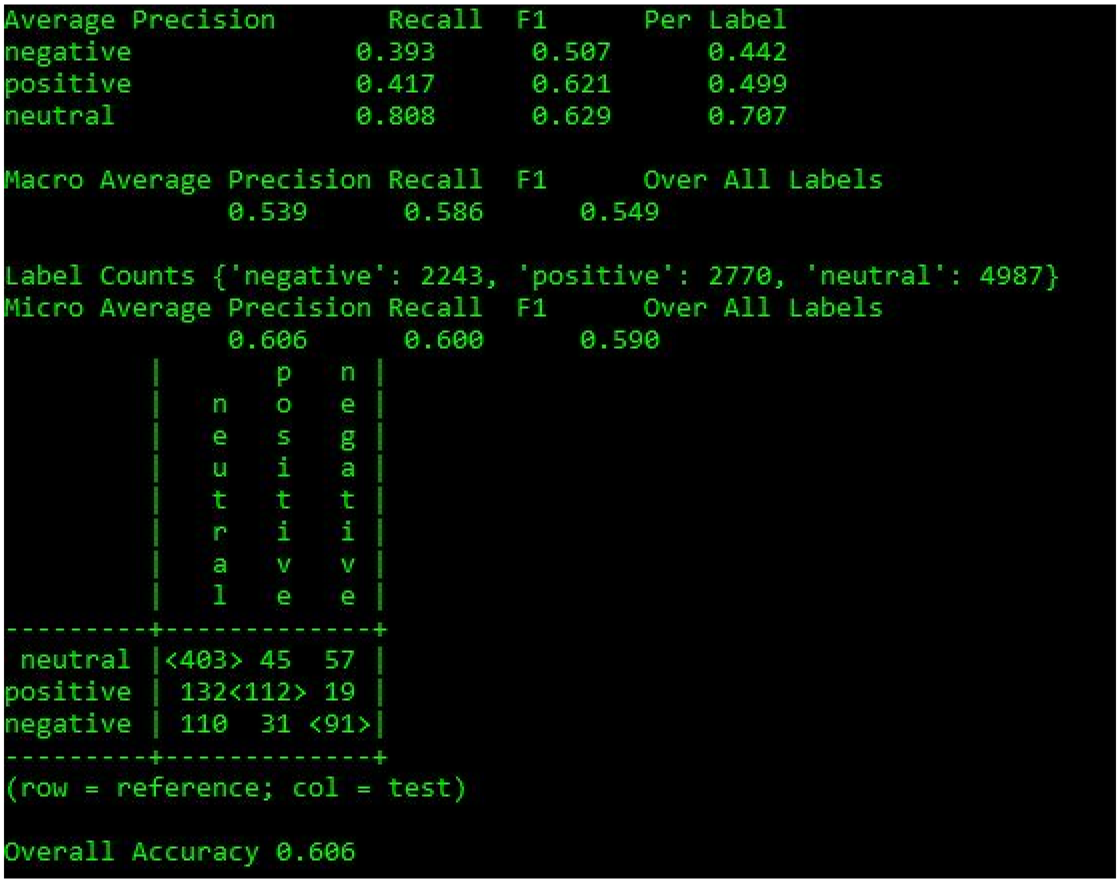
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
negative              0.393      0.507      0.442
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707

Macro Average Precision Recall          F1      Over All Labels
              0.539      0.586      0.549

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall          F1      Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

Implementing bigrams with a chi square score, did not change the classification accuracy at all. We will experiment to see if the bigrams with a raw frequency measure have an affect on the classifier.

STARTING POINT ACCURACY



```
Average Precision       Recall  F1      Per Label
negative              0.393      0.507      0.442
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707

Macro Average Precision Recall  F1      Over All Labels
              0.539      0.586      0.549

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall  F1      Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

WHAT WAS IMPLEMENTED

Bigrams were implemented to the python file with binned data. It is important to note that when creating bigrams, you cannot remove stopwords as then the bigrams would not be accurate. This attempt used the raw frequency measure.

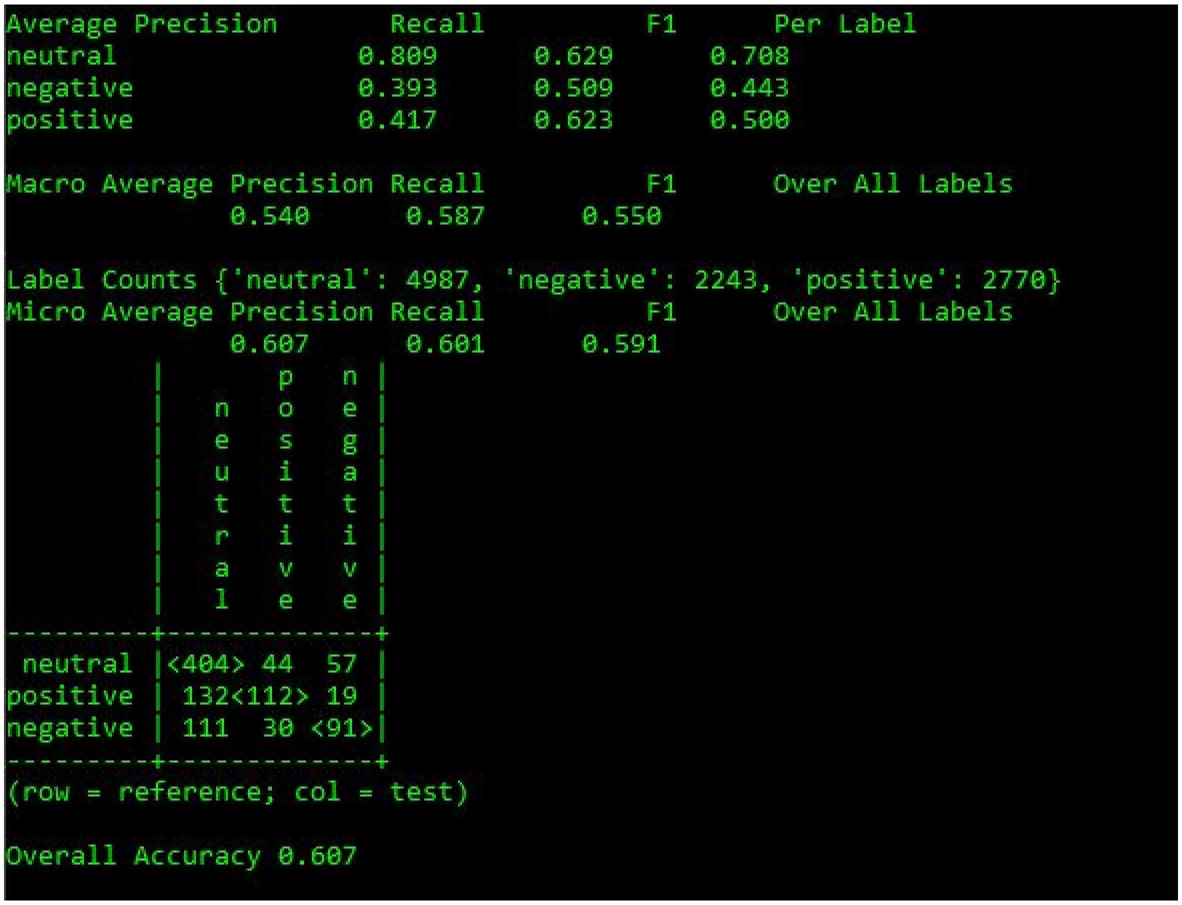
CODE

bigram_features = finder.nbest(bigram_measures.raw_freq, 500)

IN ENGLISH

This line evaluates the bigrams using the pmi method and returns the top 500 bigrams based on their raw frequency.

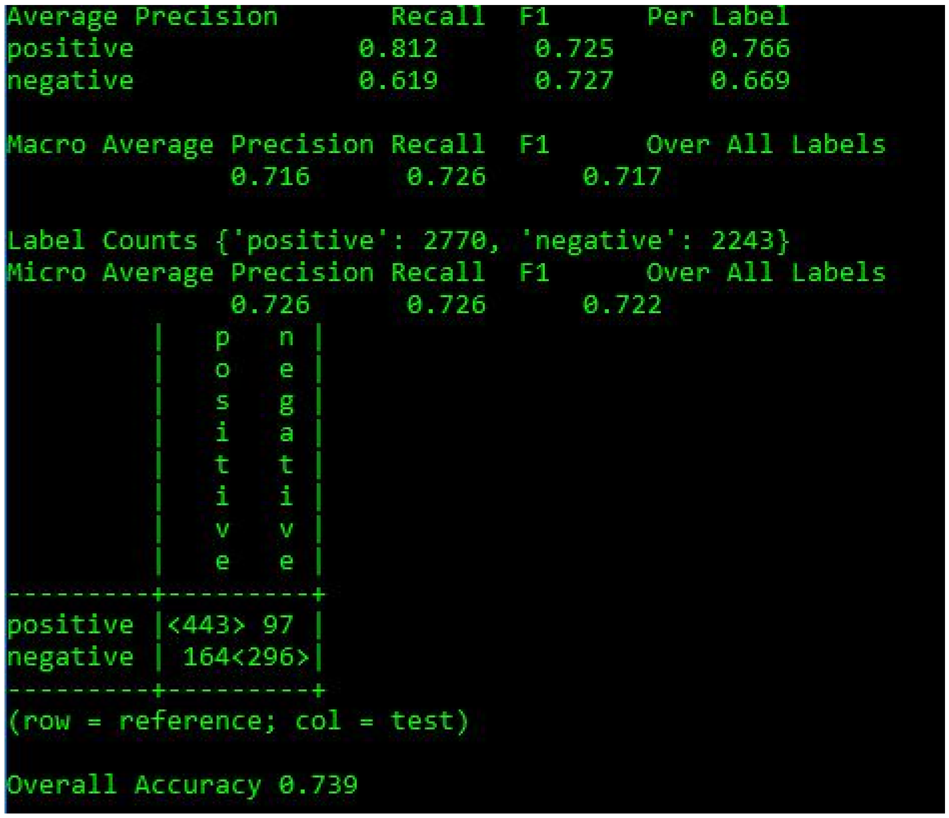
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
neutral               0.809      0.629      0.708
negative              0.393      0.509      0.443
positive              0.417      0.623      0.500

Macro Average Precision Recall          F1      Over All Labels
              0.540      0.587      0.550

Label Counts {'neutral': 4987, 'negative': 2243, 'positive': 2770}
Micro Average Precision Recall          F1      Over All Labels
              0.607      0.601      0.591
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<404> 44  57 |
positive | 132<112> 19 |
negative | 111  30 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.607
```

Bigrams with a raw frequency measure, did impact the overall accuracy very slightly. The original accuracy was 60.6% and the new overall accuracy is 60.7%. With the raw frequency we were able to correctly classify one addition neutral movie review, which was previously classified as positive.

STARTING POINT ACCURACY - NO NEUTRAL

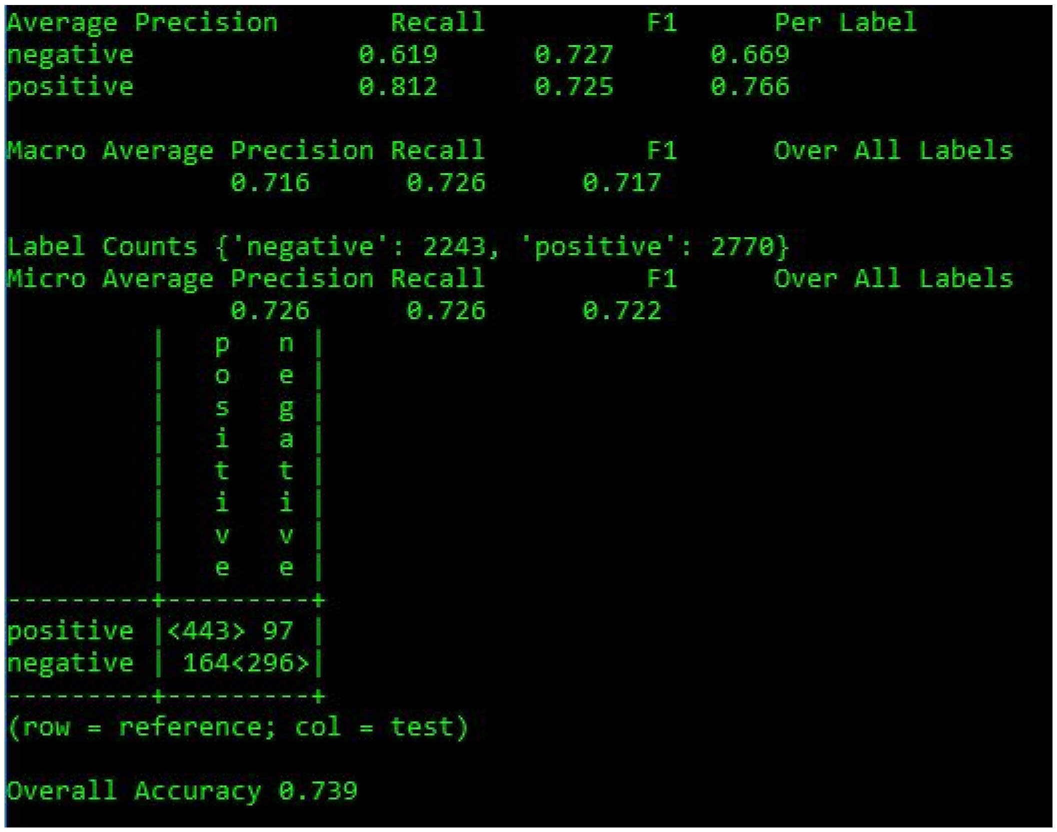


```
Average Precision       Recall  F1      Per Label
positive              0.812      0.725      0.766
negative              0.619      0.727      0.669

Macro Average Precision Recall  F1      Over All Labels
              0.716      0.726      0.717

Label Counts {'positive': 2770, 'negative': 2243}
Micro Average Precision Recall  F1      Over All Labels
              0.726      0.726      0.722
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<443> 97 |
negative | 164<296>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.739
```

WHAT WAS IMPLEMENTED

Bigrams were implemented to the python file neutral removed. The steps to implement bigrams are described above.

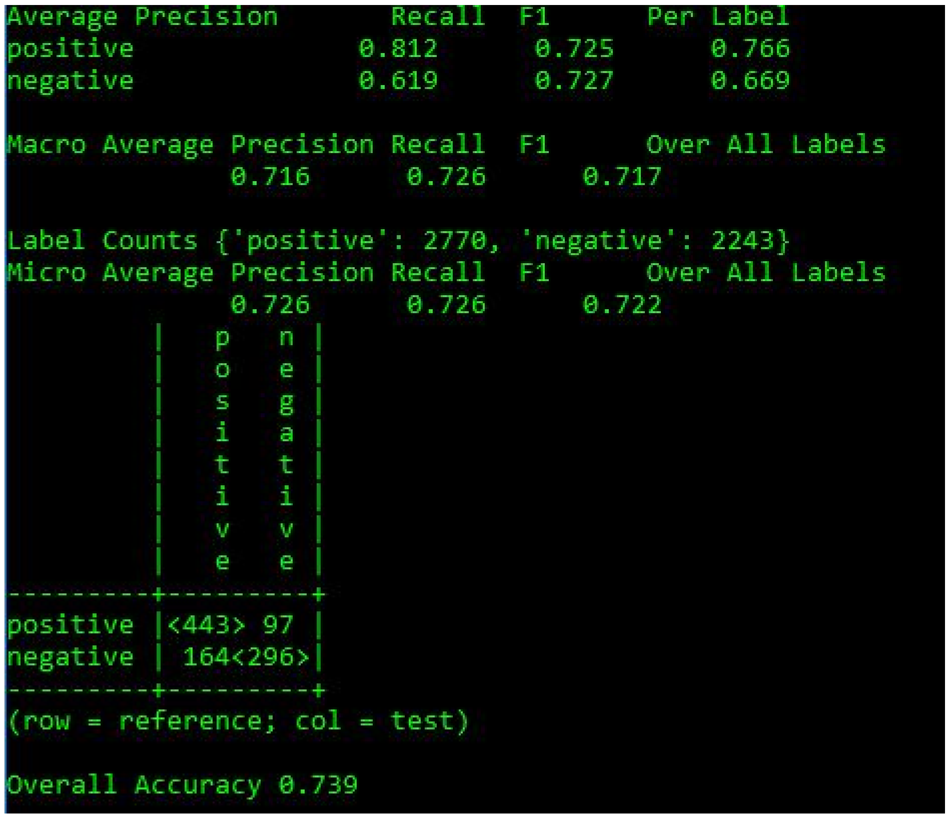
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
negative             0.619      0.727      0.669
positive             0.812      0.725      0.766

Macro Average Precision Recall          F1      Over All Labels
              0.716      0.726      0.717

Label Counts {'negative': 2243, 'positive': 2770}
Micro Average Precision Recall          F1      Over All Labels
              0.726      0.726      0.722
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<443> 97 |
negative | 164<296>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.739
```

Bigrams with a pmi measure had no effect on the classifier for the negative and positive reviews. We will attempt this with bigrams with a raw frequency measure.

STARTING POINT ACCURACY - NO NEUTRAL

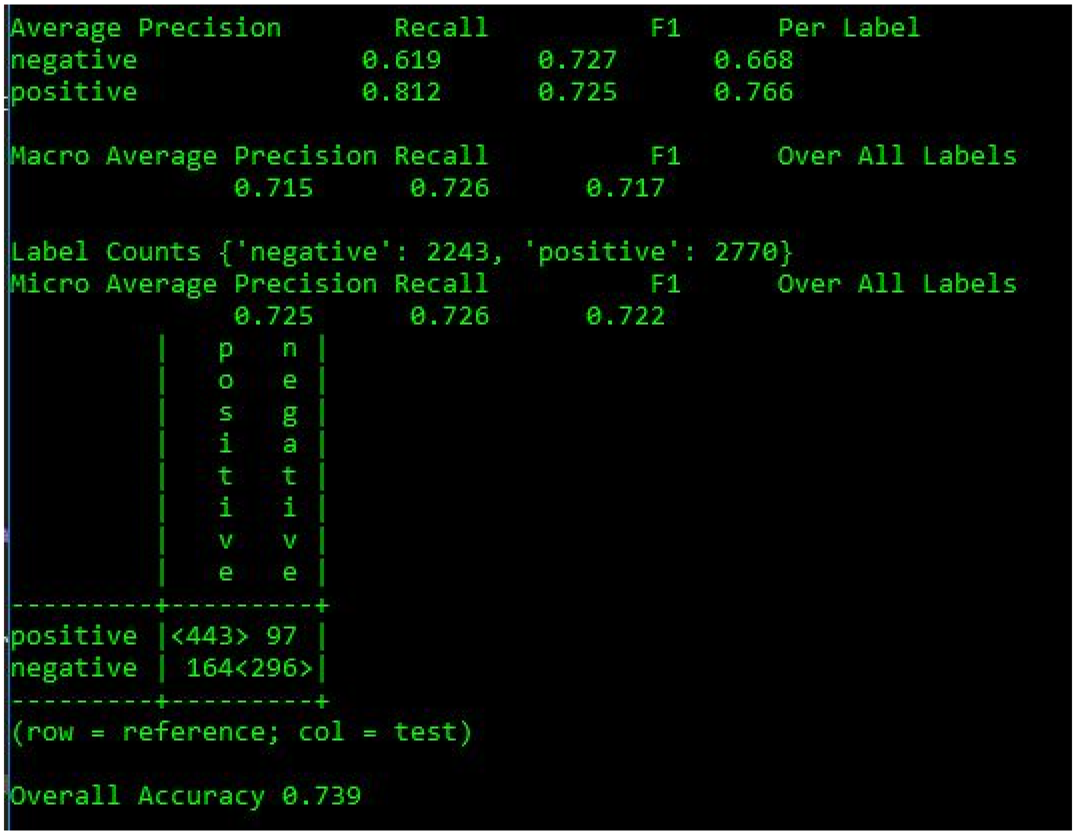


```
Average Precision       Recall  F1      Per Label
positive              0.812      0.725      0.766
negative              0.619      0.727      0.669

Macro Average Precision Recall  F1      Over All Labels
              0.716      0.726      0.717

Label Counts {'positive': 2770, 'negative': 2243}
Micro Average Precision Recall  F1      Over All Labels
              0.726      0.726      0.722
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<443> 97 |
negative | 164<296>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.739
```

WHAT WAS IMPLEMENTED

Bigrams were implemented to the python file neutral removed. The steps to implement bigrams are described above.

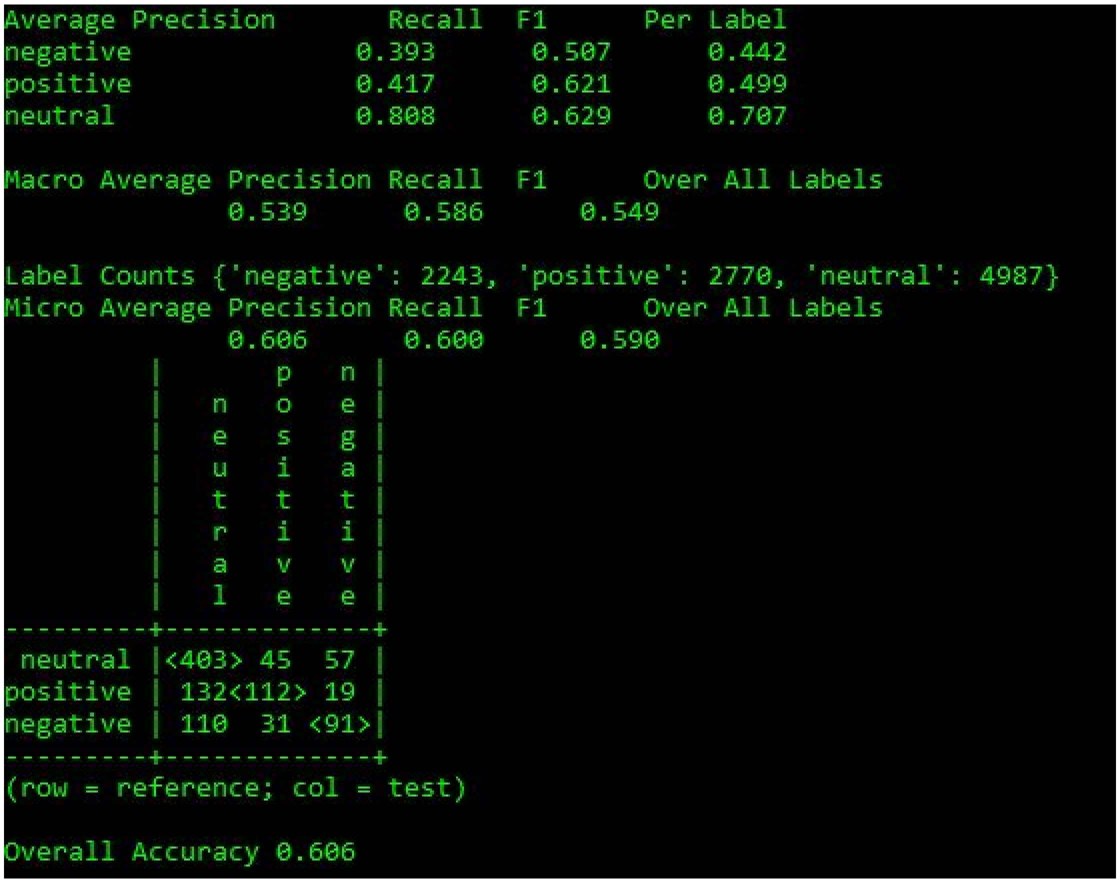
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
negative              0.619      0.727      0.668
positive              0.812      0.725      0.766

Macro Average Precision Recall          F1      Over All Labels
              0.715      0.726      0.717

Label Counts {'negative': 2243, 'positive': 2770}
Micro Average Precision Recall          F1      Over All Labels
              0.725      0.726      0.722
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<443> 97 |
negative | 164<296>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.739
```

Bigrams with a raw frequency measure had a minimal effect on the positive and negative review classifier. The F1 per label decreased by one one-thousandth for negative reviews. The Macro average precision decreased by one one-thousandth, as well. The overall accuracy remained the same. Since this is a balanced dataset, we are discussing macro averages and not micro averages.

STARTING POINT ACCURACY



```
Average Precision       Recall  F1      Per Label
negative              0.393      0.507      0.442
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707

Macro Average Precision Recall  F1      Over All Labels
              0.539      0.586      0.549

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall  F1      Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

WHAT WAS IMPLEMENTED

A score of the negative and positive words were included in the classifier.

CODE

```
def readSubjectivity(path):
  flexicon = open(path, 'r')
  sldict = { }
  for line in flexicon:
    fields = line.split()
    strength = fields[0].split("=")[1]
    word = fields[2].split("=")[1]
    posTag = fields[3].split("=")[1]
    stemmed = fields[4].split("=")[1]
    polarity = fields[5].split("=")[1]
    if (stemmed == 'y'):
```

```
    isStemmed = True
  else:
    isStemmed = False
  sldict[word] = [strength, posTag, isStemmed, polarity]
return sldict
```

IN ENGLISH

This function creates three arrays: poslist, neutrallist, & neglist. It goes through all of the words and appends each array depending on if the word is positive, negative or neutral.

CODE

```
SLpath = "./SentimentLexicons/subjclueslen1-HLTEMNLP05.tff"
SL = readSubjectivity(SLpath)
```

IN ENGLISH

This code uses the read_subkectivity_three_types function defined above to read in the subjclueslen1-HLTEMNLP05.tff file.

CODE

```
def SL_features(document, word_features, SL):
  document_words = set(document)
  features = {}

  for word in word_features:
          features['V_{}'.format(word)] = (word in document_words)
  # count variables for the 4 classes of subjectivity
  weakPos = 0
  strongPos = 0
  weakNeg = 0
  strongNeg = 0
  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['positivecount'] = weakPos + (5 * strongPos)
          features['negativecount'] = weakNeg + (5 * strongNeg)
  return features
```

IN ENGLISH

Then there is another loop that goes through every single word and creates a count of words that are weak positive, strong positive, weak negative, or strong negative. Weak positive and negative words are only counted once, however strong positive and negative words are given more weight and counted 5 times. The function ultimately produces features that include a positive count and a negative count for each review.

CODE

```
SL_featuresets = [(SL_features(d, word_features, SL), c) for (d, c) in docs]
```

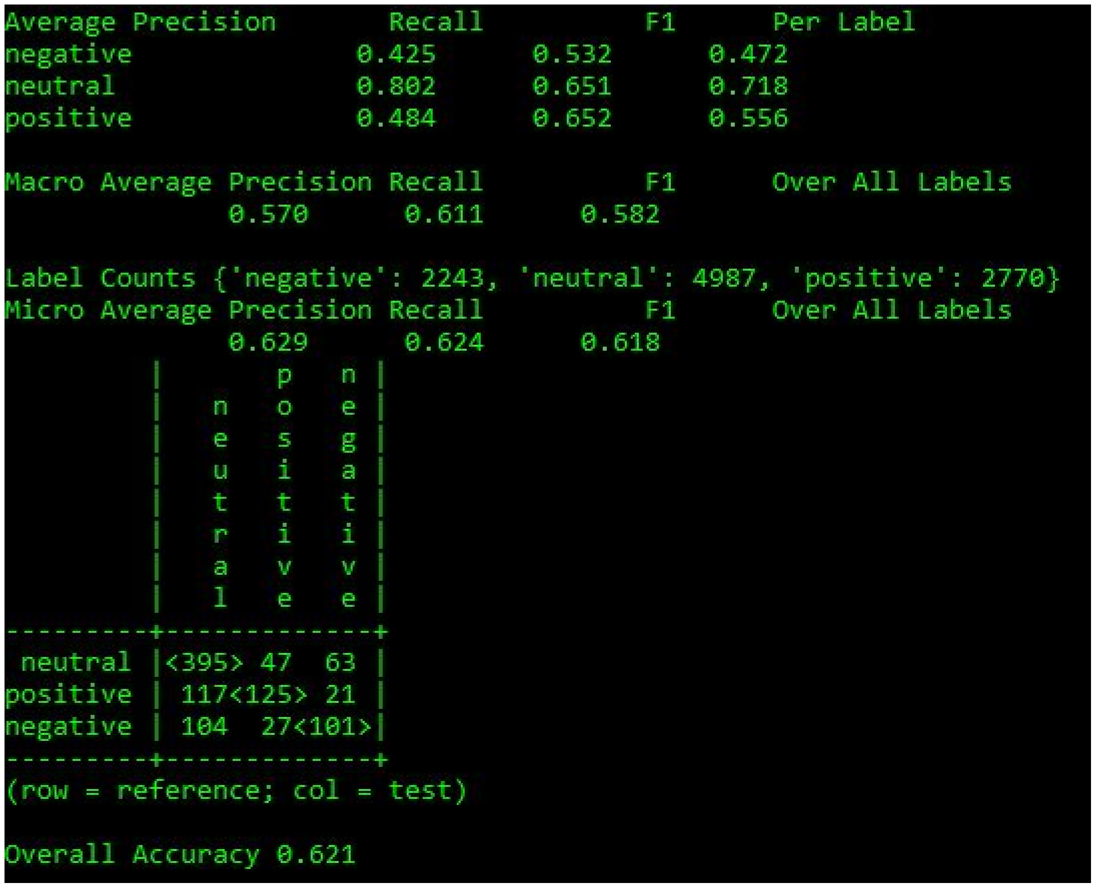
IN ENGLISH

Calls the SL_features function that was defined above. The SL_features is generating an array. Each item in the array has both an object of features and the sentiment. The object of features contains every word in the phrases followed by true or false, depending on if the word is represented in that specific phrase, it also contains a positive and negative word count followed by the sentiment.

REASON

We felt that classification would be assisted with a list of positive and negative word scores. We believed that positive reviews would have a higher positive word score than neutral and negative reviews. Likewise, that negative reviews would have a higher negative word score than neutral and positive reviews.

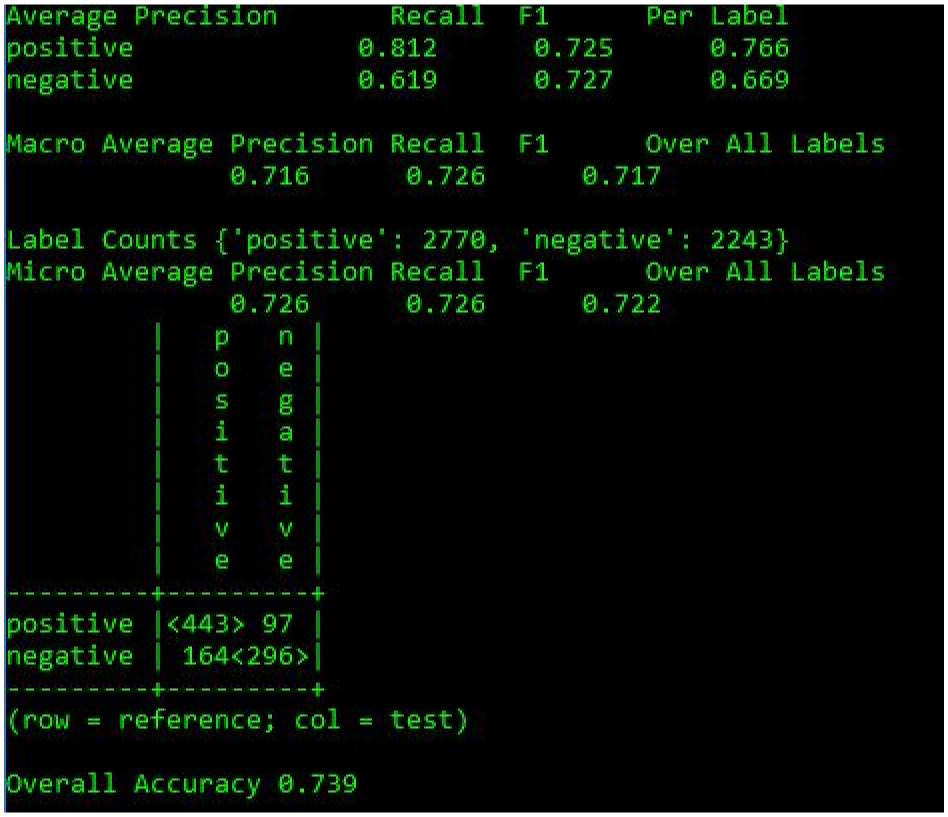
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
negative              0.425      0.532      0.472
neutral               0.802      0.651      0.718
positive              0.484      0.652      0.556

Macro Average Precision Recall          F1      Over All Labels
              0.570      0.611      0.582

Label Counts {'negative': 2243, 'neutral': 4987, 'positive': 2770}
Micro Average Precision Recall          F1      Over All Labels
              0.629      0.624      0.618
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<395> 47  63 |
positive | 117<125> 21 |
negative | 104  27<101>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.621
```

The inclusion of word subjectivity positively influence our accuracy. Precision for neutral slightly decreased, but precision for negative and positive increased. The micro precision, recall and F1 all increased. We were able to successfully classify 13 more positive reviews and 10 more negative reviews. We only correctly classified 395 neutral reviews compared to the 403 neutral reviews that were correctly classified without subjectivity. This is acceptable as the classifier is now classifying more reviews and negative and positive and not all reviews as neutral.

STARTING POINT ACCURACY - NO NEUTRAL

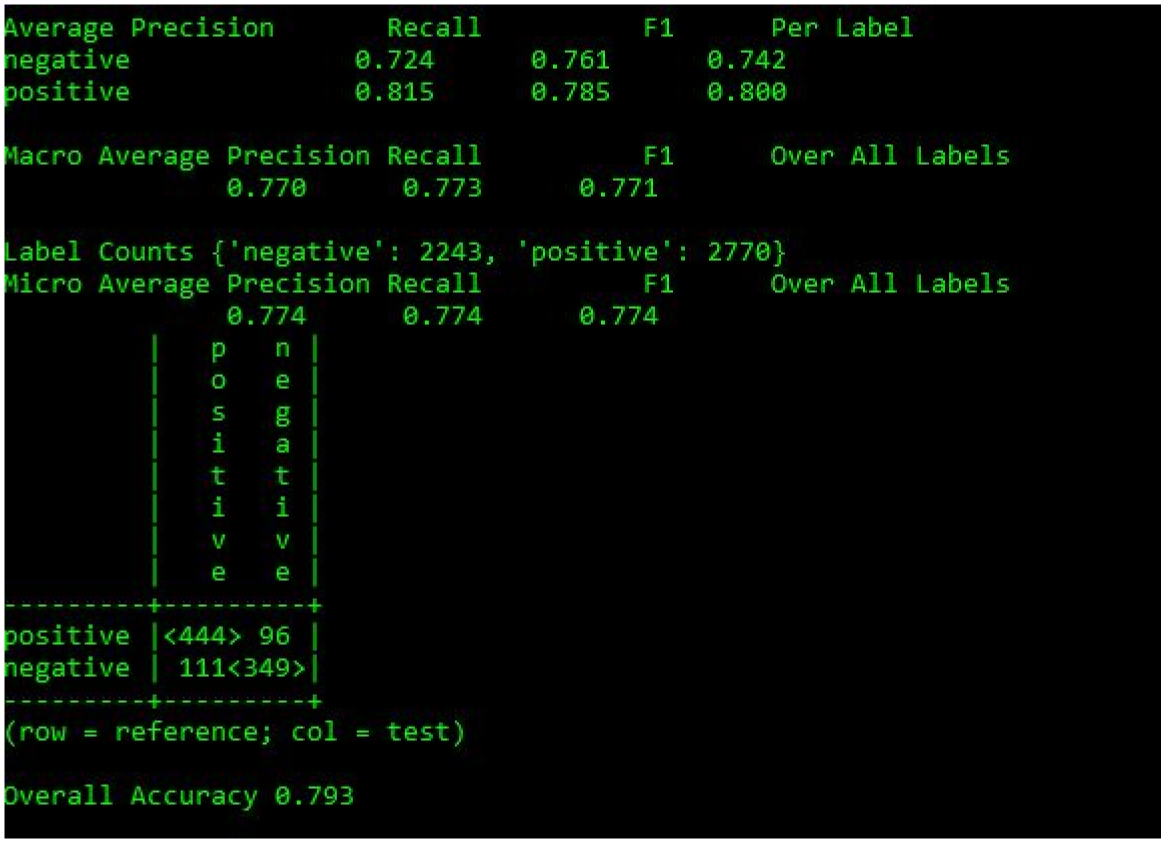


```
Average Precision       Recall  F1      Per Label
positive              0.812      0.725      0.766
negative              0.619      0.727      0.669

Macro Average Precision Recall  F1      Over All Labels
              0.716      0.726      0.717

Label Counts {'positive': 2770, 'negative': 2243}
Micro Average Precision Recall  F1      Over All Labels
              0.726      0.726      0.722
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<443> 97 |
negative | 164<296>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.739
```

WHAT WAS IMPLEMENTED

Sentiment for the negative and positive file

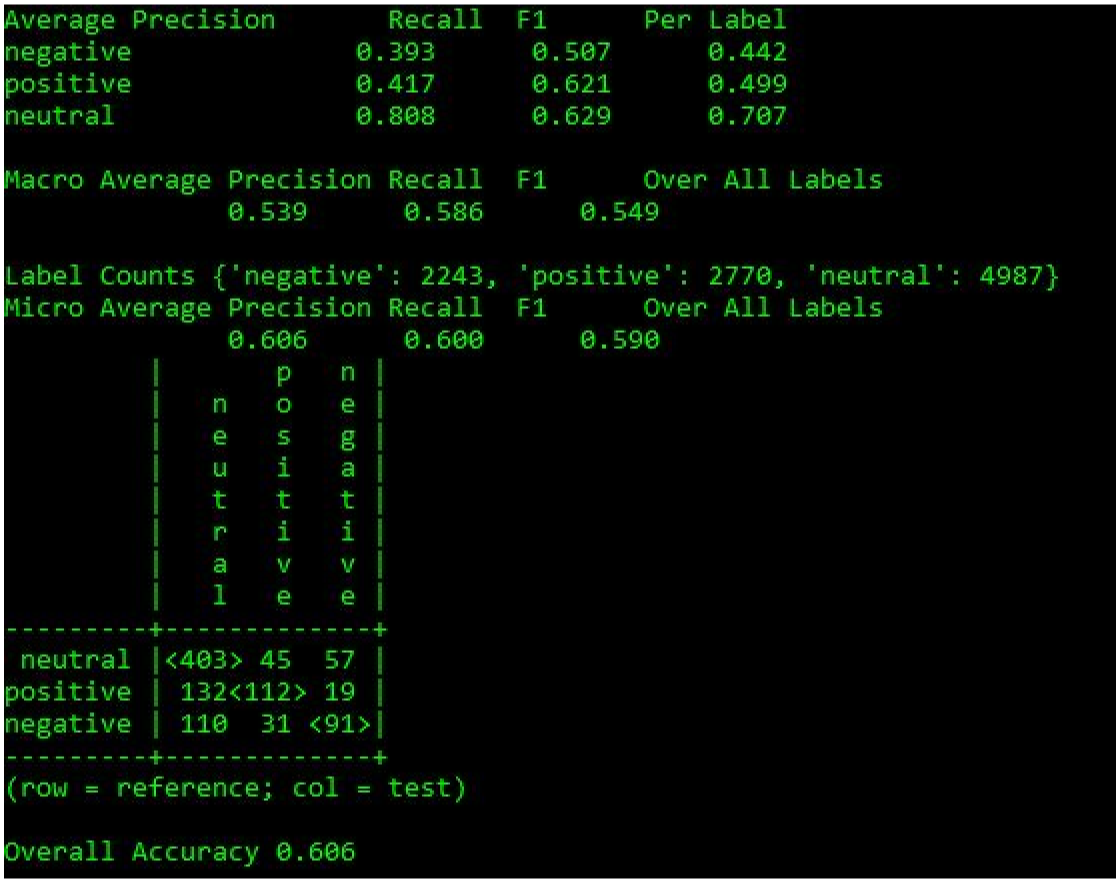
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
negative              0.724      0.761      0.742
positive              0.815      0.785      0.800

Macro Average Precision Recall          F1      Over All Labels
              0.770      0.773      0.771

Label Counts {'negative': 2243, 'positive': 2770}
Micro Average Precision Recall          F1      Over All Labels
              0.774      0.774      0.774
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<444> 96 |
negative | 111<349>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.793
```

The inclusion of subjectivity greatly increased precision for negative reviews. It also increased the F1 for both negative and positive reviews. We were able to correctly classify one addition positive movie review and 53 negative movie reviews. Word subjectivity was extremely beneficial in our accuracy for positive and negative movie reviews.

STARTING POINT ACCURACY



```
Average Precision       Recall  F1      Per Label
negative              0.393      0.507      0.442
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707

Macro Average Precision Recall  F1      Over All Labels
              0.539      0.586      0.549

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall  F1      Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

WHAT WAS IMPLEMENTED

This attempt combined negation and bigrams together.

CODE

```
def Not_features(document, word_features, bigram_features, negationwords):
  document_words = set(document)
  document_bigrams = nltk.bigrams(document)
  features = {}
  for word in word_features:
        features['V_{}'.format(word)] = False
        features['V_NOT{}'.format(word)] = False
  for bigram in bigram_features:
        features['B_{}_{}'.format(bigram[0], bigram[1])] = (bigram in
        document_bigrams)
        #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) or
        (word.endswith("n't"))):
           i += 1
           features['V_NOT{}'.format(document[i])] = (document[i] in word_features)
        else:
           features['V_{}'.format(word)] = (word in word_features)
return features
```

IN ENGLISH

The code above is a combination of the code that was used in the negation experiment and the bigram experiment. The code is combined in the Not_features function, which is comprised of multiple loops.

CODE

```
NOT_featuresets = [(Not_features(d, word_features, bigram_features,
negationwords), c) for (d, c) in docs]
```

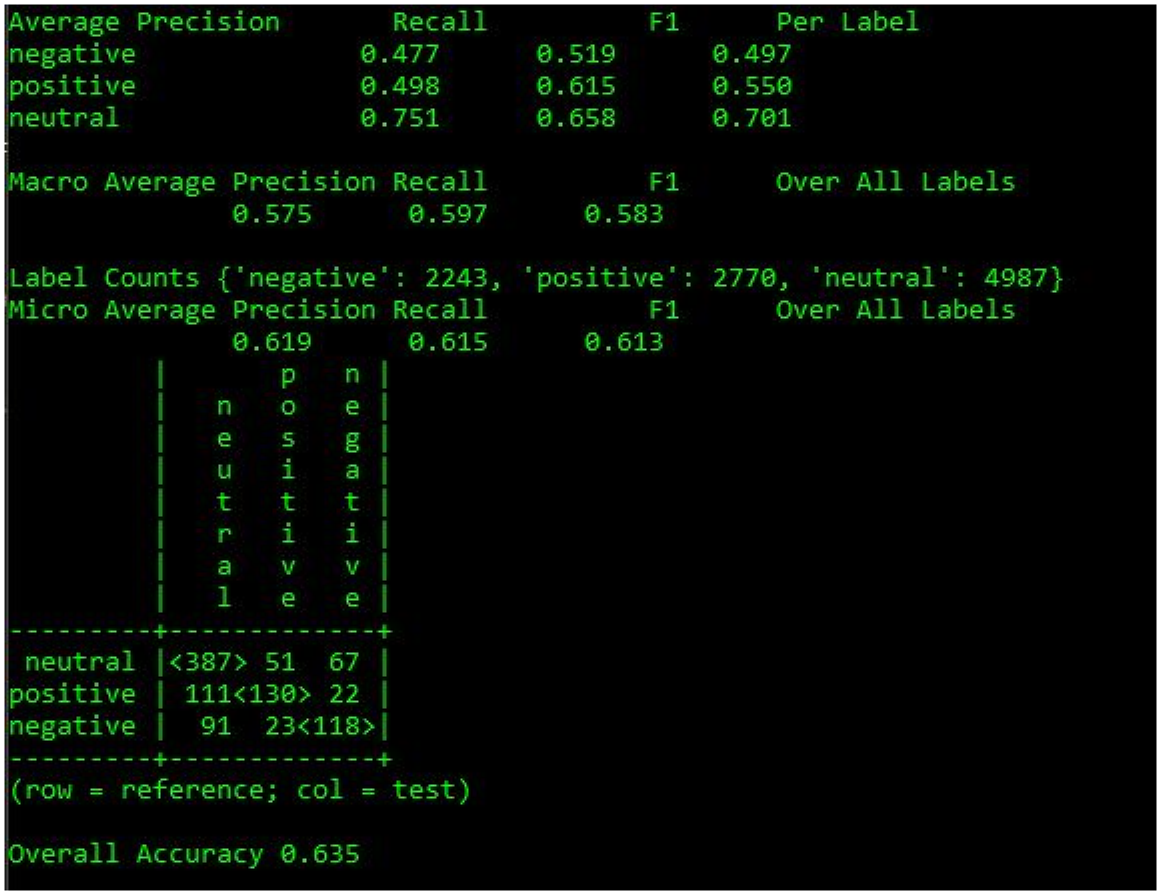
IN ENGLISH

This calls the NOT_featuresets function defined above, that includes word_features, bigram_features and negationwords. The function creates an array that includes the word_features, bigram_features, negationwords, and the sentiment for each review.

REASON

Initially, we believed that combining negation with bigrams might prove fruitful. However, after running the bigrams and seeing little if no improvement, we are not sure if the combination will offer higher results than negation alone.

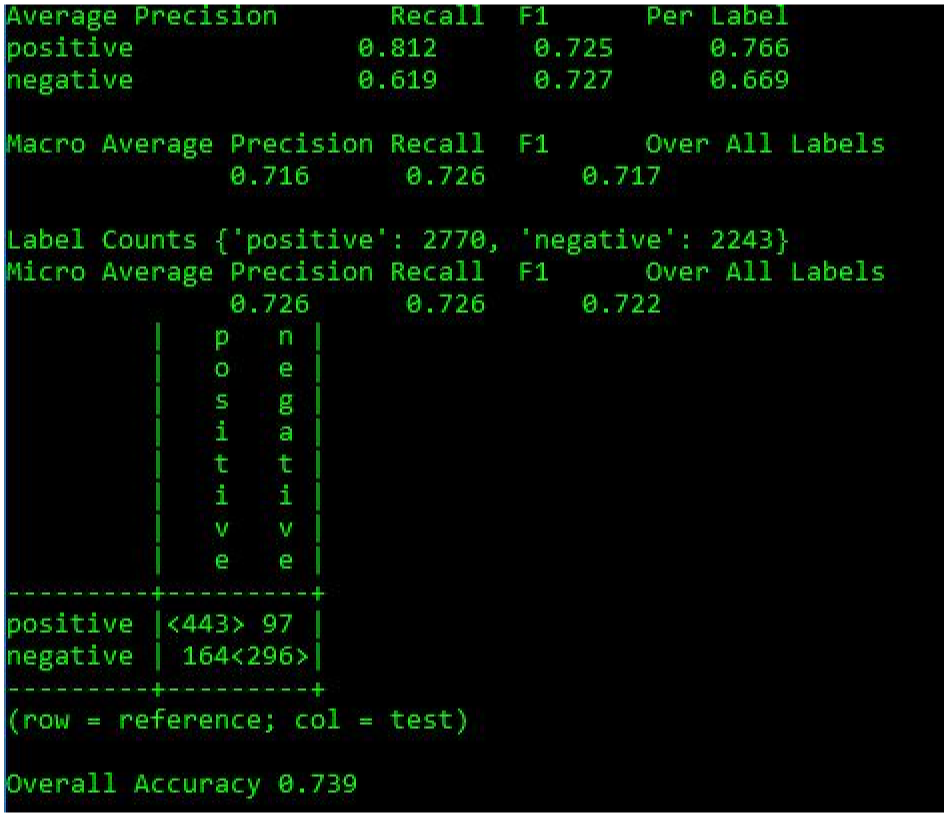
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
negative             0.477      0.519      0.497
positive             0.498      0.615      0.550
neutral              0.751      0.658      0.701

Macro Average Precision Recall          F1      Over All Labels
              0.575      0.597      0.583

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall          F1      Over All Labels
              0.619      0.615      0.613
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<387> 51  67 |
positive | 111<130> 22 |
negative |  91  23<118>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.635
```

The combination of bigrams with a raw frequency measure and negation increased the overall accuracy. However, when compared to the cross evaluation, confusion matrix and overall accuracy from the negation attempt, the accuracy score only increased by one one-thousandth from .634 to .635. We were able to successfully classify a review that was wrongly classified as negative and correctly classify it as neutral. Also, one negative review was previously classified as positive, but in this attempt was classified as neutral. This means that the classifier is getting closer to correctly classifying it. The Micro averages did not change at all for precision, recall and F1 from the initial negation accuracy levels.

STARTING POINT ACCURACY - NO NEUTRAL

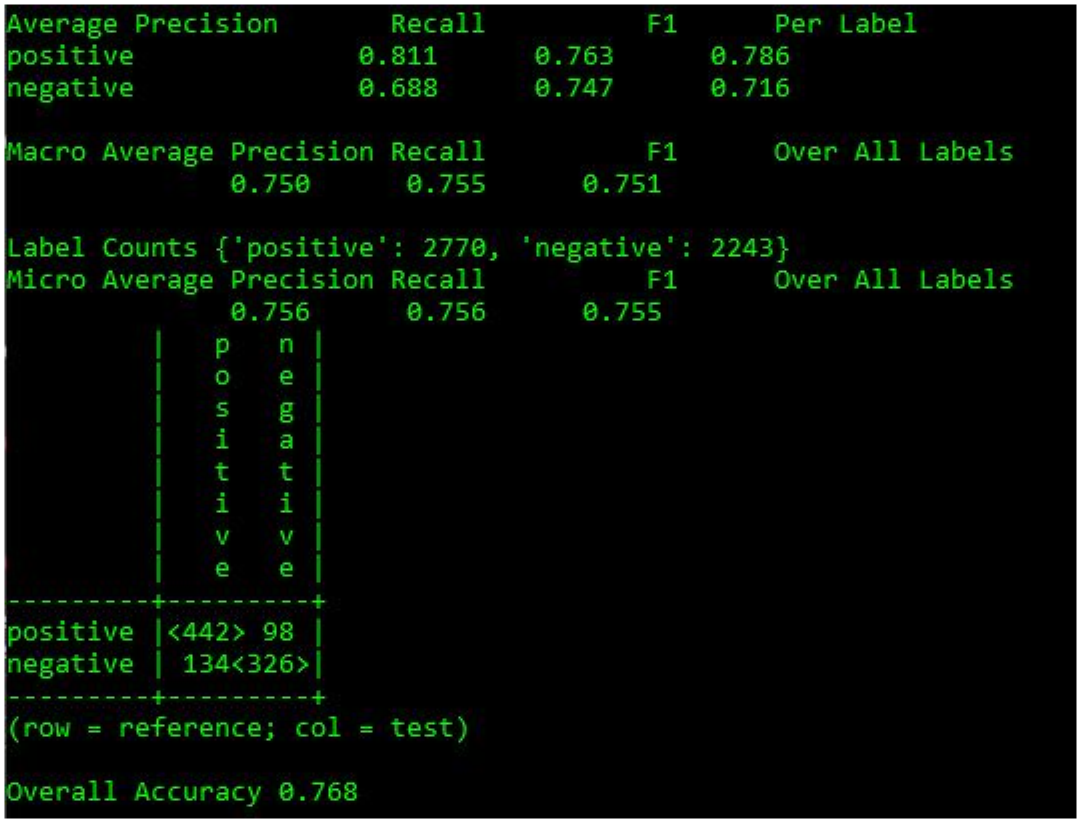


```
Average Precision       Recall  F1      Per Label
positive              0.812      0.725      0.766
negative              0.619      0.727      0.669

Macro Average Precision Recall  F1      Over All Labels
              0.716      0.726      0.717

Label Counts {'positive': 2770, 'negative': 2243}
Micro Average Precision Recall  F1      Over All Labels
              0.726      0.726      0.722
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<443> 97 |
negative | 164<296>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.739
```

WHAT WAS IMPLEMENTED

Bigrams with negation for the file with neutral removed.

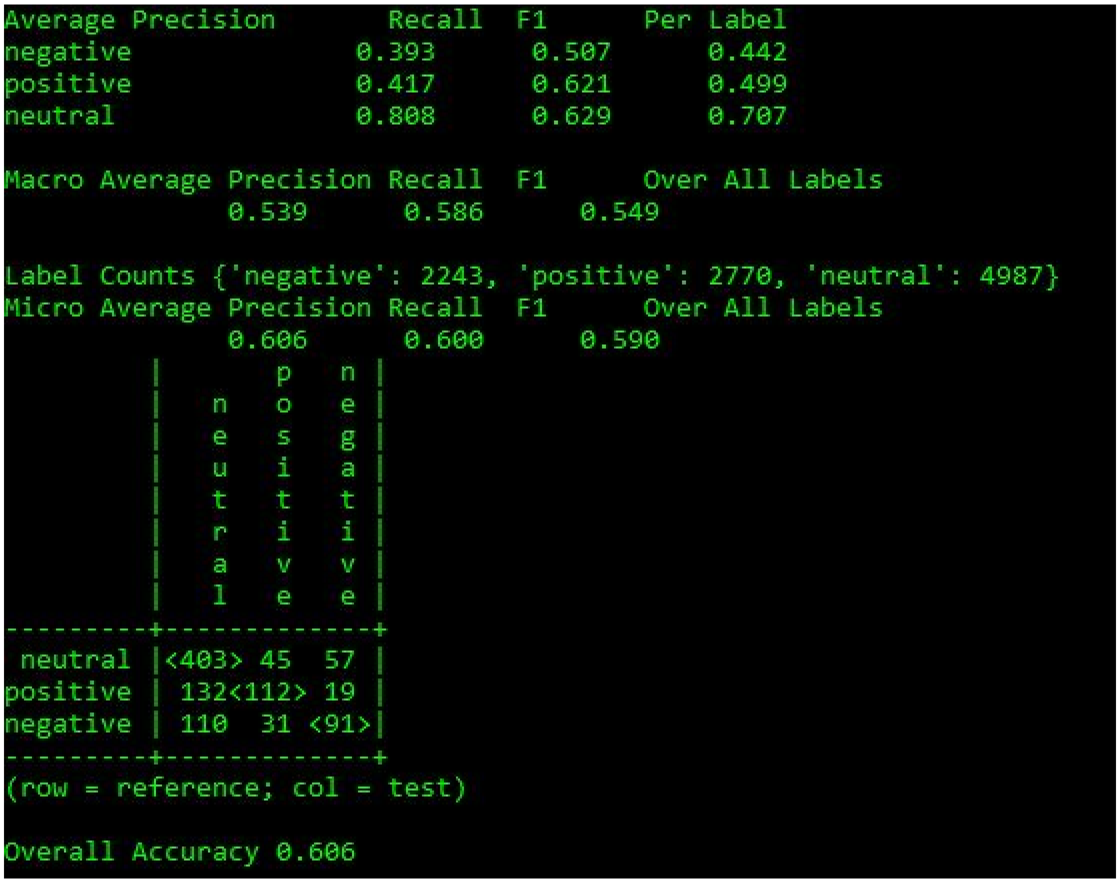
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
positive              0.811      0.763      0.786
negative              0.688      0.747      0.716

Macro Average Precision Recall          F1      Over All Labels
              0.750      0.755      0.751

Label Counts {'positive': 2770, 'negative': 2243}
Micro Average Precision Recall          F1      Over All Labels
              0.756      0.756      0.755
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<442> 98 |
negative | 134<326>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.768
```

When comparing the combination of bigrams and negation to the baseline, it appears that it helped with precision, recall, F1 and overall accuracy, however, this was mainly due to negation and not bigrams.

STARTING POINT ACCURACY



```
Average Precision       Recall  F1      Per Label
negative              0.393      0.507      0.442
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707

Macro Average Precision Recall  F1      Over All Labels
              0.539      0.586      0.549

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall  F1      Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

WHAT WAS IMPLEMENTED

This attempt combined negation and removed stopwords.

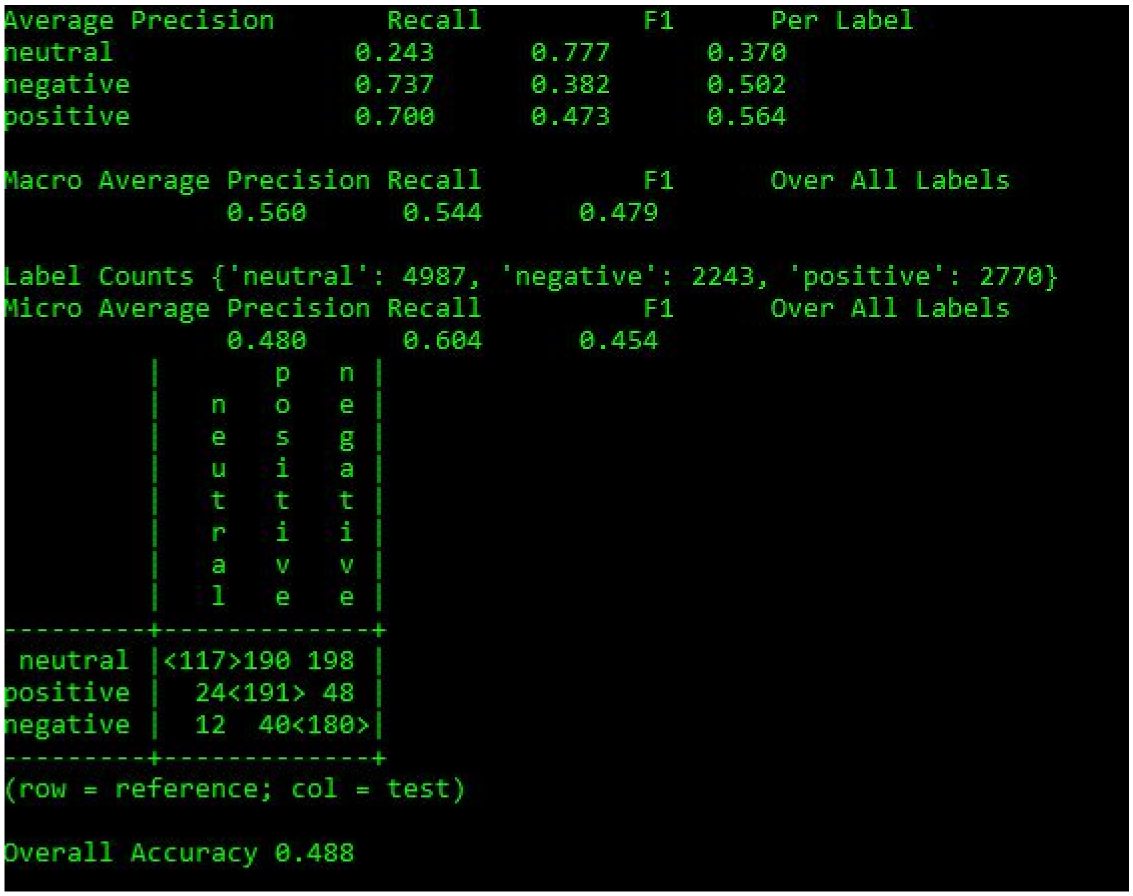
CODE

```
def Not_features(document, word_features, negationwords):
  features = {}
  for word in word_features:
        features['contains(V_{})'.format(word)] = False
        features['contains(V_NOT{})'.format(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) or (word.endswith("n't"))):
        i += 1
        features['V_NOT{}'.format(document[i])] = (document[i] in word_features)
        else:
        features['V_{}'.format(word)] = (word in word_features)
  return features
```

IN ENGLISH

This is the same code from the negation experiment. In fact, no new code was implemented. The only difference is that the stopwords were removed from the all_words_list prior to running the Not_features function on the data.

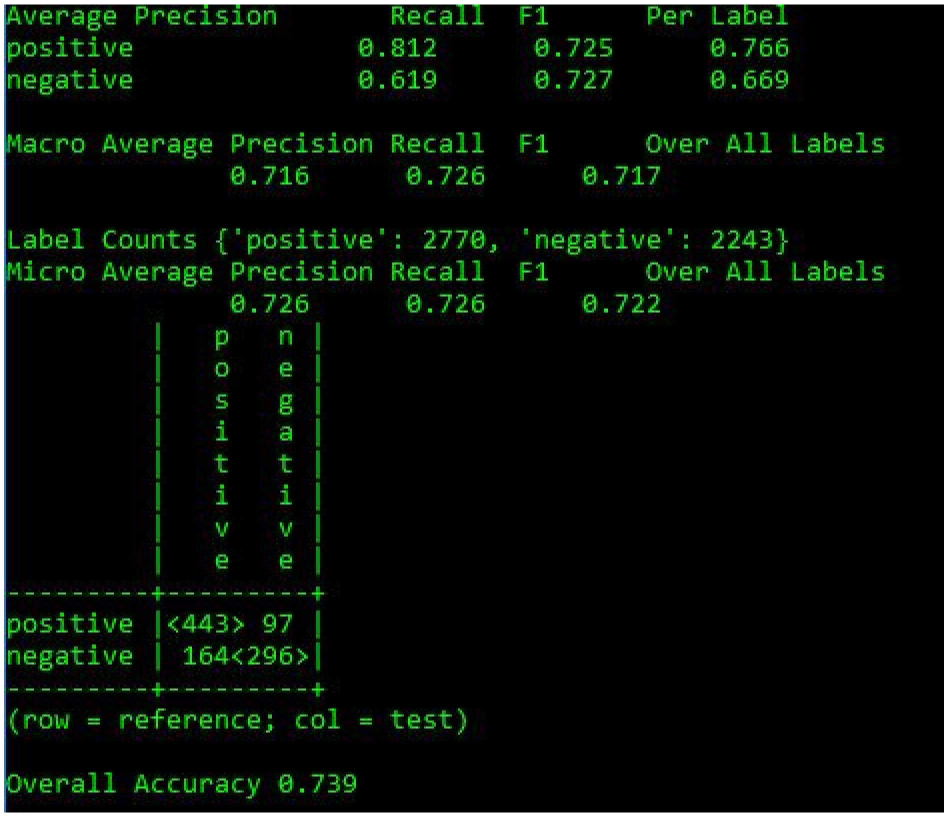
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
neutral               0.243      0.777      0.370
negative              0.737      0.382      0.502
positive              0.700      0.473      0.564

Macro Average Precision Recall          F1      Over All Labels
              0.560      0.544      0.479

Label Counts {'neutral': 4987, 'negative': 2243, 'positive': 2770}
Micro Average Precision Recall          F1      Over All Labels
              0.480      0.604      0.454
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<117>190 198 |
positive |  24<191> 48 |
negative |  12  40<180>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.488
```

This experiment had an adverse effect on the overall accuracy. However, it was able to correctly classify more positive and negative movie reviews. It incorrectly classified neutral reviews mainly as positive or negative. Neutral has a low precision, but the precision for negative and positive greatly increased. Which for movie reviews, we believe that precision is more important than recall, since false negatives are not as serious. Which would be different for spam detection.

STARTING POINT ACCURACY - NO NEUTRAL

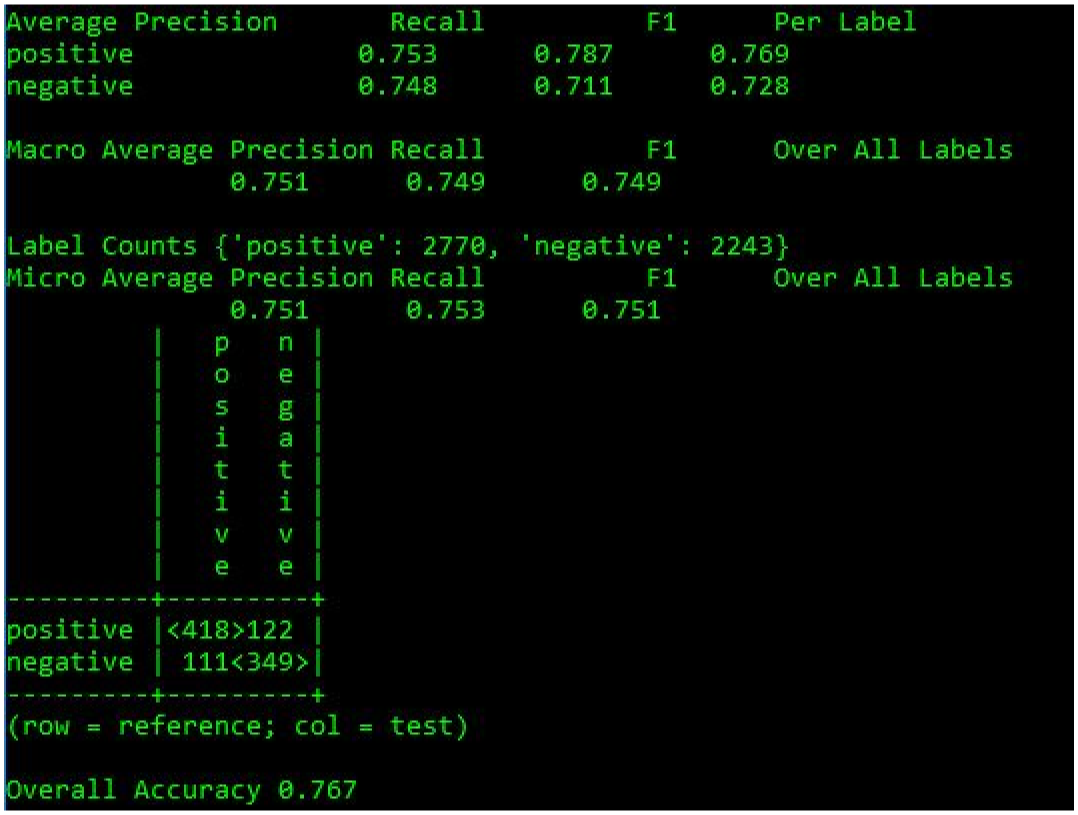


```
Average Precision       Recall  F1      Per Label
positive              0.812      0.725      0.766
negative              0.619      0.727      0.669

Macro Average Precision Recall  F1      Over All Labels
              0.716      0.726      0.717

Label Counts {'positive': 2770, 'negative': 2243}
Micro Average Precision Recall  F1      Over All Labels
              0.726      0.726      0.722
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<443> 97 |
negative | 164<296>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.739
```

WHAT WAS IMPLEMENTED

Negation with stopwords removed for the file with neutral removed.

NEW ACCURACY

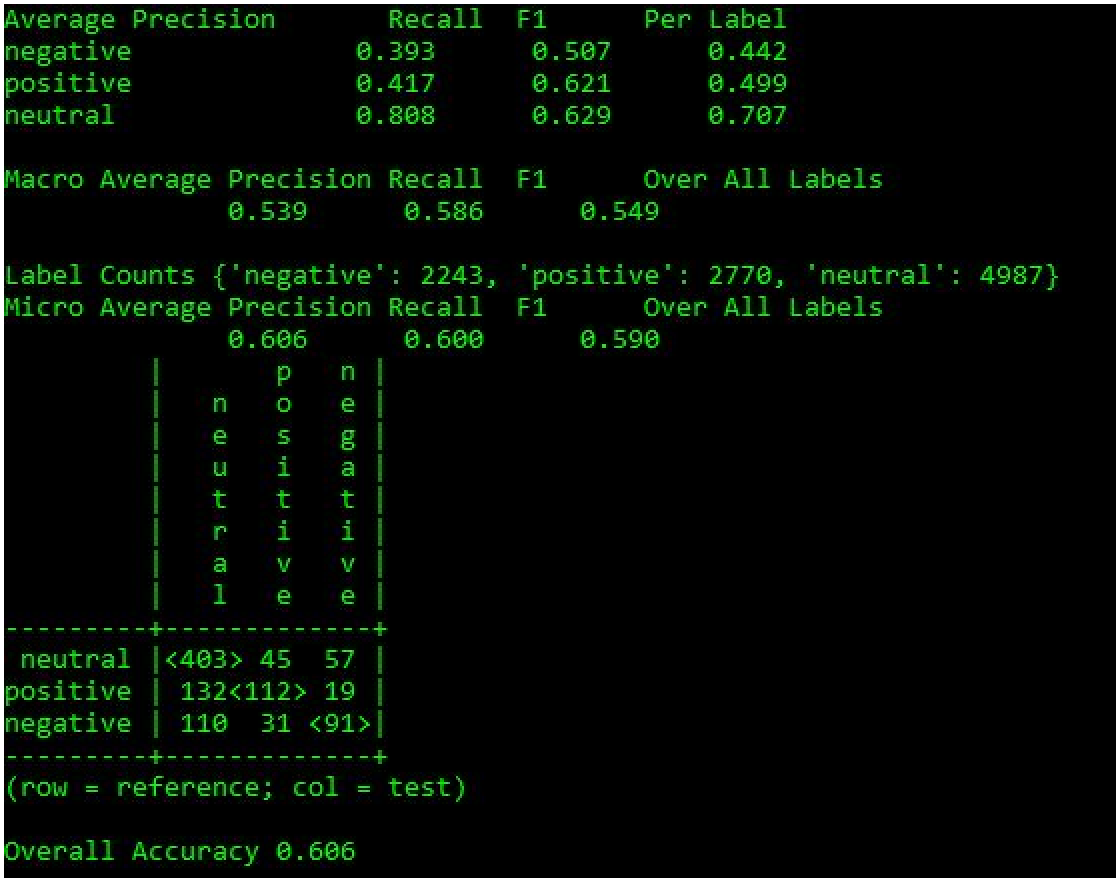


```
Average Precision       Recall          F1      Per Label
positive              0.753      0.787      0.769
negative              0.748      0.711      0.728

Macro Average Precision Recall          F1      Over All Labels
              0.751      0.749      0.749

Label Counts {'positive': 2770, 'negative': 2243}
Micro Average Precision Recall          F1      Over All Labels
              0.751      0.753      0.751
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<418>122 |
negative | 111<349>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.767
```

At first glance, it appears as if the negation with stopwords removed benefited our accuracy levels. We lost precision with negative reviews, but were able to successfully classify 53 more negative reviews than the baseline. However, when compared to the attempt with only negation a different story is told. Our precision for positive decreased, but our precision for negative increased. We successfully classified 24 less positive reviews, but correctly classified 23 more negative reviews with stopwords removed, than by negation alone. This model does a better job classifying negative reviews and a slightly worse job classifying positive reviews than the model with negation only.

ALL TOGETHER NOW

STARTING POINT ACCURACY



```
Average Precision       Recall  F1      Per Label
negative              0.393      0.507      0.442
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707

Macro Average Precision Recall  F1      Over All Labels
              0.539      0.586      0.549

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall  F1      Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

WHAT WAS IMPLEMENTED

A COMBINATION OF ***ALL THE FEATURE FUNCTIONS!!***

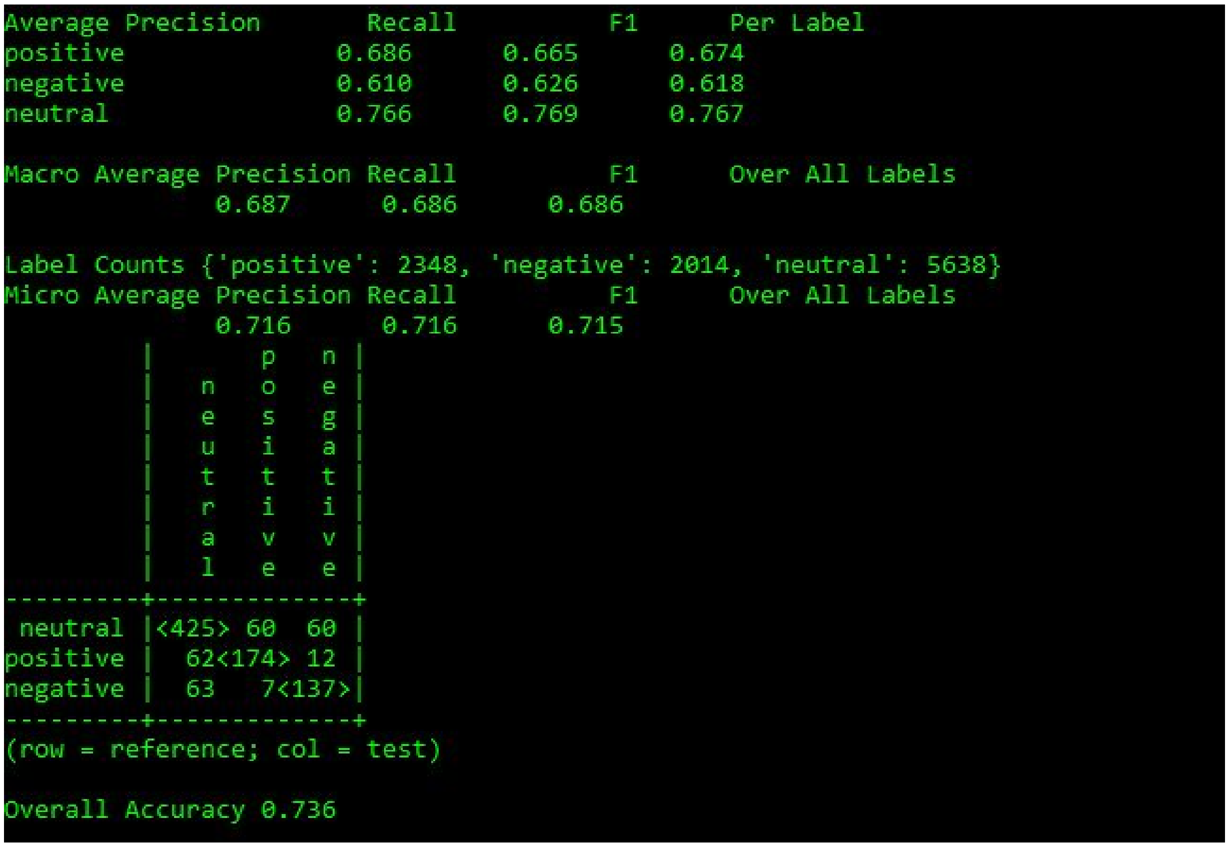
CODE

```
def combined_document_features(document, word_features, SL, SL2,
bigram_features, negationwords):
  document_words = set(document)
  document_bigrams = nltk.bigrams(document)
  features = {}
  # SUBJECTIVITY: Getting strength and polarity from readSubjectivity
  # (this function gives an object that includes polarity AND strength,
much more useful)
  # Adds ['positiveStrengthCount'] & ['negativeStrengthCount'] to our
features object
  weakPos = 0
  strongPos = 0
  weakNeg = 0
  strongNeg = 0
  for word in document_words:
    if word in SL2:
      strength, posTag, isStemmed, polarity = SL2[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['positiveStrengthCount'] = (2 * weakPos) + (5 * strongPos)
      features['negativeStrengthCount'] = (2 * weakNeg) + (5 * strongNeg)
  # SUBJECTIVITY: Getting word counts from read_subjectivity_three_types
  # (this function gives an array, significantly less useful)
  # Adds ['positivecount'] & ['negativecount'] & ['neutralcount'] to our
features object
  posword = 0
  neutword = 0
  negword = 0
  for word in document_words:
    if word in SL[0]:
      posword += 1
    if word in SL[1]:
      neutword += 1
    if word in SL[2]:
      negword += 1
    features['positivecount'] = posword
    features['neutralcount'] = neutword
    features['negativecount'] = negword
  # NEGATION WORDS: This is a combination of the original
"document_features" function
  # And an if/else to deal with negation words.
  # Adds V_ and V_NOT to our features
  for word in word_features:
    features['V_{}'.format(word)] = False
    features['V_NOT{}'.format(word)] = False
  for word in word_features:
    for i in range(0, len(document)):
      word = document[i]
      if ((i + 1) < len(document)) and ((word in negationwords) or
(word.endswith("n't"))):
          i += 1
          features['V_NOT{}'.format(document[i])] = (document[i] in
word_features)
      else:
          features['V_{}'.format(word)] = (word in word_features)
  # BIGRAMS: this gets the bigrams
  # Adds B_ to our features
  for bigram in bigram_features:
    features['B_{}_{}'.format(bigram[0], bigram[1])] = (bigram in
document_bigrams)
  return features
```

IN ENGLISH

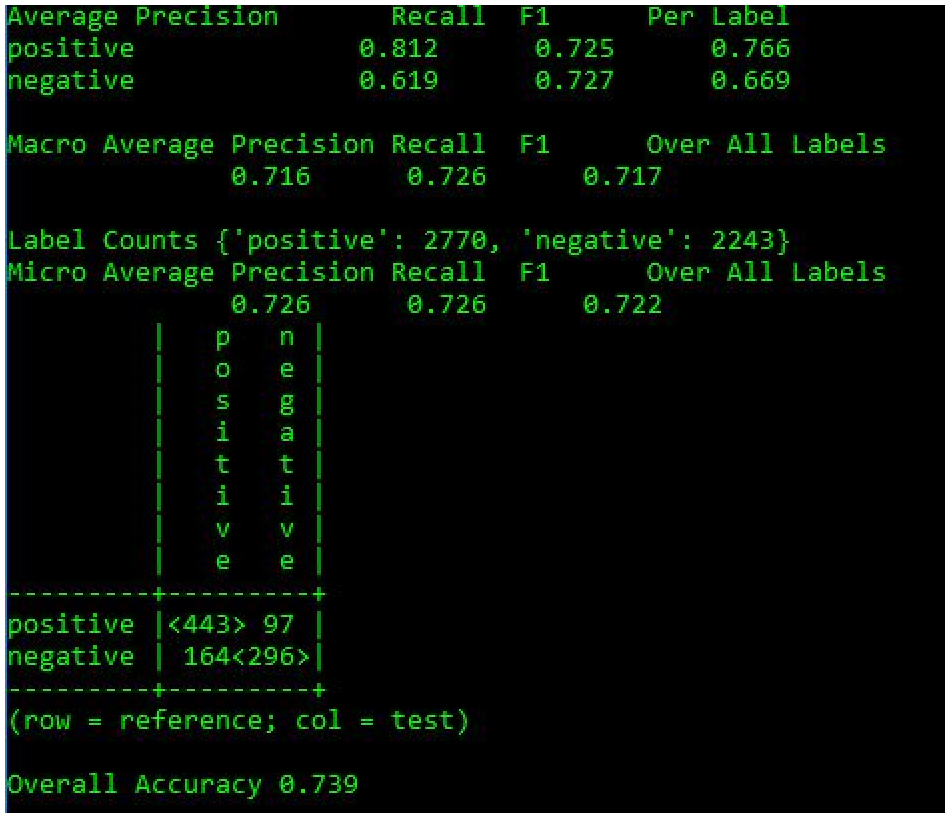
We piled everything we learned from our previous experiments into one giant function. See comments above.

NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
positive              0.686      0.665      0.674
negative              0.610      0.626      0.618
neutral               0.766      0.769      0.767

Macro Average Precision Recall          F1      Over All Labels
              0.687      0.686      0.686

Label Counts {'positive': 2348, 'negative': 2014, 'neutral': 5638}
Micro Average Precision Recall          F1      Over All Labels
              0.716      0.716      0.715
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<425> 60  60 |
positive |  62<174> 12 |
negative |  63   7<137>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.736
```

STARTING POINT ACCURACY - NO NEUTRAL

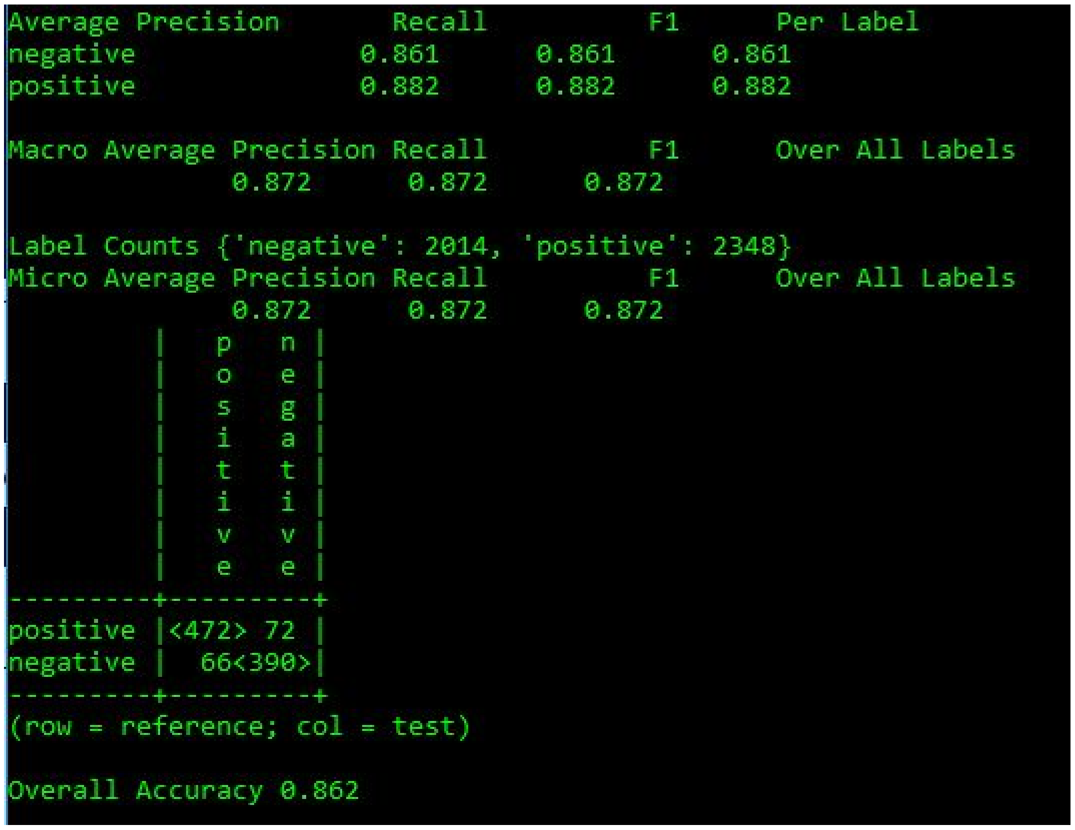


```
Average Precision       Recall  F1       Per Label
positive              0.812      0.725      0.766
negative              0.619      0.727      0.669

Macro Average Precision Recall  F1       Over All Labels
              0.716      0.726      0.717

Label Counts {'positive': 2770, 'negative': 2243}
Micro Average Precision Recall  F1       Over All Labels
              0.726      0.726      0.722
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<443> 97 |
negative | 164<296>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.739
```

WHAT WAS IMPLEMENTED

All the feature functions with neutral removed.

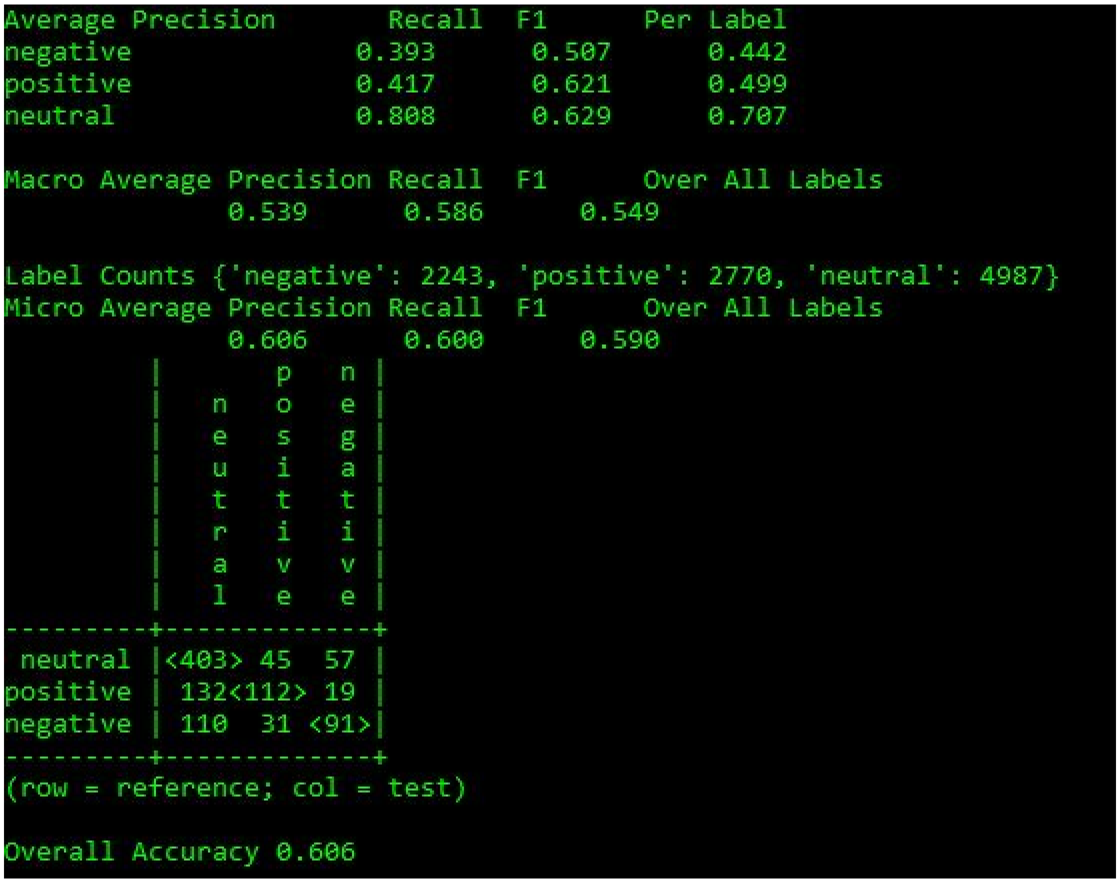
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
negative              0.861      0.861      0.861
positive              0.882      0.882      0.882

Macro Average Precision Recall          F1      Over All Labels
              0.872      0.872      0.872

Label Counts {'negative': 2014, 'positive': 2348}
Micro Average Precision Recall          F1      Over All Labels
              0.872      0.872      0.872
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<472> 72 |
negative |  66<390>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.862
```

Combining all of the features greatly improved our accuracy. We were able to correctly classify 94 additional negative movie reviews and 29 additional positive movie reviews. Our accuracies for precision, recall, and F1 all increased as did our overall accuracy.

STARTING POINT ACCURACY

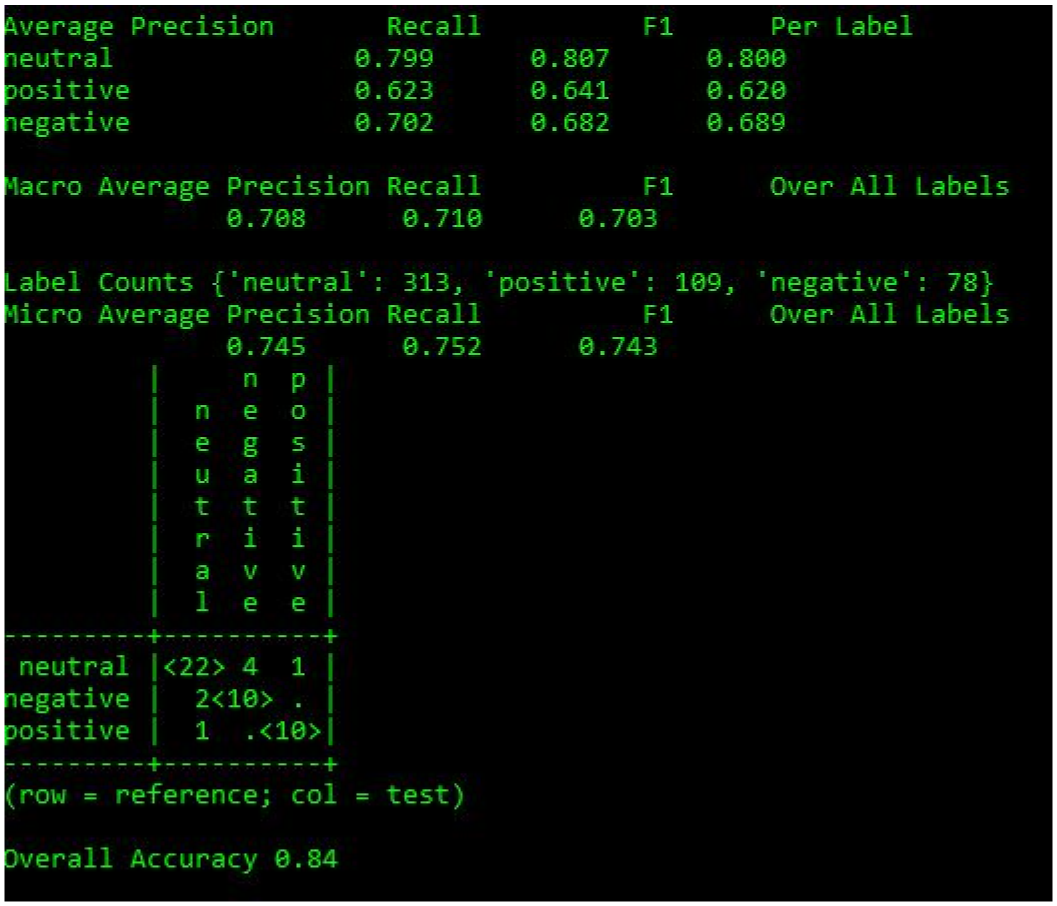


```
Average Precision       Recall  F1      Per Label
negative              0.393      0.507      0.442
positive              0.417      0.621      0.499
neutral               0.808      0.629      0.707

Macro Average Precision Recall  F1      Over All Labels
              0.539      0.586      0.549

Label Counts {'negative': 2243, 'positive': 2770, 'neutral': 4987}
Micro Average Precision Recall  F1      Over All Labels
              0.606      0.600      0.590
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<403> 45  57 |
positive | 132<112> 19 |
negative | 110  31 <91>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.606
```

WHAT WAS IMPLEMENTED

Everything was compiled as stated above, but this time we experimented with sample size and dropped our random sample down to 500 instead of 10,000

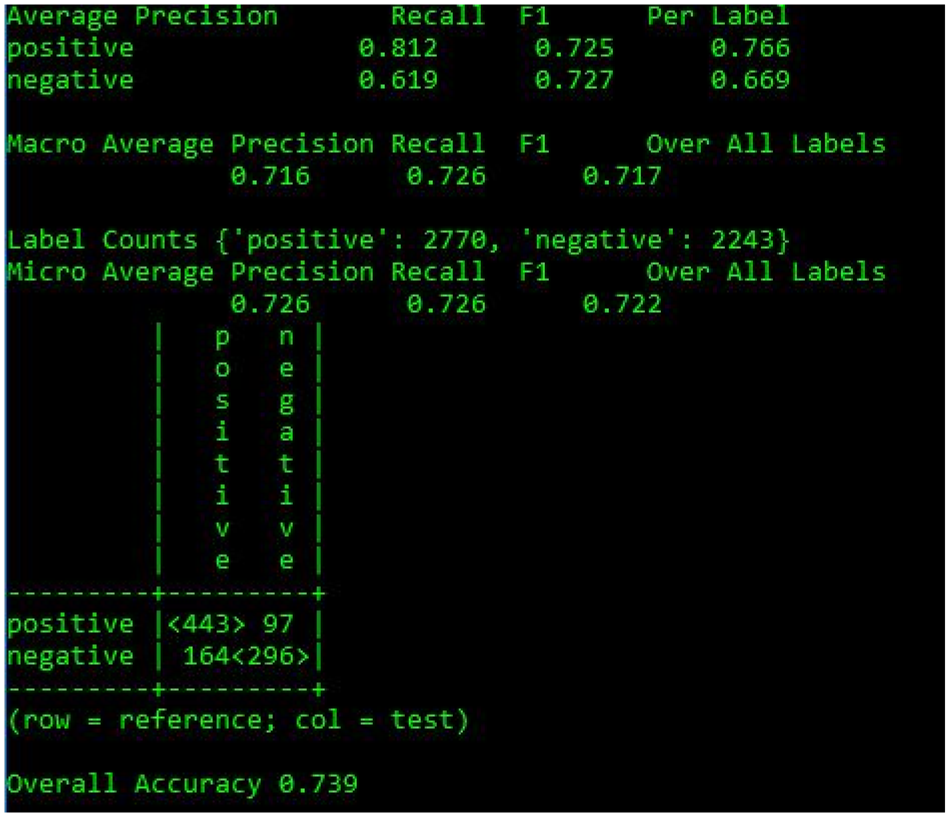
NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
neutral               0.799      0.807      0.800
positive              0.623      0.641      0.620
negative              0.702      0.682      0.689

Macro Average Precision Recall          F1      Over All Labels
              0.708      0.710      0.703

Label Counts {'neutral': 313, 'positive': 109, 'negative': 78}
Micro Average Precision Recall          F1      Over All Labels
              0.745      0.752      0.743
         |     n  p |
         |  n  e  o |
         |  e  g  s |
         |  u  a  i |
         |  t  t  t |
         |  r  i  i |
         |  a  v  v |
         |  l  e  e |
---------+----------+
 neutral |<22> 4  1 |
negative |  2<10> . |
positive |  1  .<10>|
---------+----------+
(row = reference; col = test)

Overall Accuracy 0.84
```

When running with a sample of 500, we were able to achieve an overall accuracy of 84%. The classifier successfully classified 10 out of 12 negative reviews, 10 out of 11 positive reviews, and 22 out of 27 neutral reviews. When the classifier misclassified negative and positive reviews it classified them as neutral. Decreasing our sample size proved to be extremely effective for this experiment.

STARTING POINT ACCURACY - NEUTRAL REMOVED

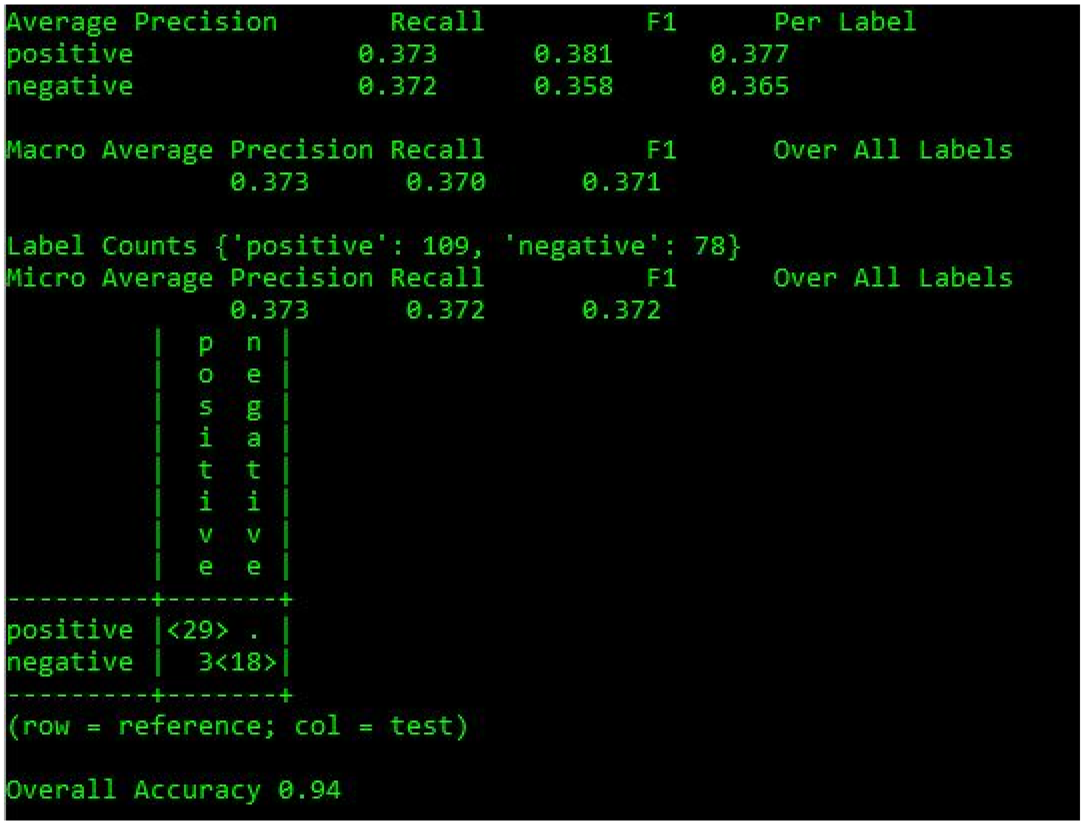


```
Average Precision       Recall  F1      Per Label
positive              0.812      0.725      0.766
negative              0.619      0.727      0.669

Macro Average Precision Recall  F1      Over All Labels
              0.716      0.726      0.717

Label Counts {'positive': 2770, 'negative': 2243}
Micro Average Precision Recall  F1      Over All Labels
              0.726      0.726      0.722
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<443> 97 |
negative | 164<296>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.739
```

WHAT WAS IMPLEMENTED

Everything was compiled as stated above, but this time we experimented with sample size and dropped our random sample down to 500 instead of 10,000

NEW ACCURACY



```
Average Precision       Recall          F1      Per Label
positive            0.373      0.381      0.377
negative            0.372      0.358      0.365

Macro Average Precision Recall          F1      Over All Labels
              0.373      0.370      0.371

Label Counts {'positive': 109, 'negative': 78}
Micro Average Precision Recall          F1      Over All Labels
              0.373      0.372      0.372
         |  p  n |
         |  o  e |
         |  s  g |
         |  i  a |
         |  t  t |
         |  i  i |
         |  v  v |
         |  e  e |
---------+-------+
positive |<29> . |
negative |  3<18>|
---------+-------+
(row = reference; col = test)

Overall Accuracy 0.94
```

When running with a sample of 500, we were able to achieve an overall accuracy of 94%. The classifier successfully classified all 29 positive reviews and 18 out of 21 negative reviews. While our overall accuracy greatly increased, our average precision, recall and F1 decreased.

EXPERIMENTS: Part Two

Testing multiple features in a combined file

MovieReviews_1.py -- our baseline

STARTING POINT ACCURACY -- 0.504

```
Average Precision Recall      F1    Per Label
2            0.950      0.538      0.687
```

```
0           0.004      0.150      0.008
1           0.032      0.267      0.057
3           0.103      0.273      0.146
4           0.000      0.000      0.000

Macro Average Precision Recall      F1    Over All Labels
            0.218      0.246      0.179

Label Counts {'2': 5086, '0': 469, '1': 1767, '3': 2095, '4': 583}
Micro Average Precision Recall      F1    Over All Labels
            0.510      0.385      0.390
  |   2   3   1   4   0 |
--+---------------------+
2 |<465> 38   1   .   . |
3 | 178 <34>  5   .   . |
1 | 144  40  <5>  .   . |
4 |  35  13   2  <.>  1 |
0 |  28   7   4   .  <.>|
--+---------------------+
(row = reference; col = test)

Overall Accuracy 0.504
```

WHAT WAS IMPLEMENTED

The only thing in this “feature set” is a count of the words in each document.

CODE

moviereviews_1.py

```
def generateFeatureSets(document):
  document_words = set(document)
  features = {}
  features['length'] = len(document_words)
  return features
```

IN ENGLISH

Seeing if document length can predict sentiment

NEW ACCURACY -- 0.504 (SAME)

```
Average Precision Recall      F1    Per Label
2           0.950      0.538      0.687
```

```
0           0.004      0.150      0.008
1           0.032      0.267      0.057
3           0.103      0.273      0.146
4           0.000      0.000      0.000

Macro Average Precision Recall      F1    Over All Labels
            0.218      0.246      0.179

Label Counts {'2': 5086, '0': 469, '1': 1767, '3': 2095, '4': 583}
Micro Average Precision Recall      F1    Over All Labels
            0.510      0.385      0.390
  |   2   3   1   4   0 |
--+---------------------+
2 |<465> 38   1   .   . |
3 | 178 <34>  5   .   . |
1 | 144  40  <5>  .   . |
4 |  35  13   2  <.>  1 |
0 |  28   7   4   .  <.>|
--+---------------------+
(row = reference; col = test)

Overall Accuracy 0.504
```

Document length alone cannot predict sentiment.

MovieReviews_2.py -- binning

STARTING POINT ACCURACY -- 0.504

```
Average Precision Recall      F1    Per Label
2           0.950      0.538      0.687
0           0.004      0.150      0.008
1           0.032      0.267      0.057
3           0.103      0.273      0.146
4           0.000      0.000      0.000

Macro Average Precision Recall      F1    Over All Labels
            0.218      0.246      0.179

Label Counts {'2': 5086, '0': 469, '1': 1767, '3': 2095, '4': 583}
Micro Average Precision Recall      F1    Over All Labels
            0.510      0.385      0.390
  |   2   3   1   4   0 |
--+----------------------+
2 |<465> 38   1    .    . |
3 | 178 <34>  5    .    . |
1 | 144  40  <5>   .    . |
4 |  35  13   2   <.>   1 |
0 |  28   7   4    .   <.>|
--+----------------------+
(row = reference; col = test)

Overall Accuracy 0.504
```

WHAT WAS IMPLEMENTED

Binning

CODE

```
  phrasedocs = []
  for phrase in phraselist:
    tokens = nltk.word_tokenize(phrase[0])
    sentiment = int(phrase[1])
    if (sentiment == 2):
      phrasedocs.append((tokens, 'neutral'))
    if ((sentiment == 0) or (sentiment == 1)):
      phrasedocs.append((tokens, 'negative'))
    if ((sentiment == 3) or (sentiment == 4)):
      phrasedocs.append((tokens, 'positive'))
```

IN ENGLISH

We went from 5 sentiment rankings to 3 sentiment rankings

NEW ACCURACY -- 0.511

```
Average Precision Recall      F1     Per Label
neutral          0.876      0.576      0.695
positive         0.178      0.365      0.237
negative         0.151      0.358      0.208

Macro Average Precision Recall      F1     Over All Labels
            0.402      0.433      0.380

Label Counts {'neutral': 5086, 'positive': 2678, 'negative': 2236}
Micro Average Precision Recall      F1     Over All Labels
```

```
            0.527      0.471      0.464
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<432> 35   37 |
positive | 173 <30> 65 |
negative | 153  26 <49>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.511
```

MovieReviews_2b.py -- binning, removing neutrals

STARTING POINT ACCURACY
0.511 (see above)

WHAT WAS IMPLEMENTED
Put neutral phrases into their own array

CODE

```
  phrasedocs = []
  neutraldocs = []
  for phrase in phraselist:
    tokens = nltk.word_tokenize(phrase[0])
    sentiment = int(phrase[1])
    if (sentiment == 2):
      # phrasedocs.append((tokens, 'neutral'))
      neutraldocs.append((tokens, 'neutral'))
    if ((sentiment == 0) or (sentiment == 1)):
      phrasedocs.append((tokens, 'negative'))
    if ((sentiment == 3) or (sentiment == 4)):
      phrasedocs.append((tokens, 'positive'))
```

IN ENGLISH

Since we only want to see if we can predict negative or positive, neutrals -- often partial phrases or single words, in this particular dataset -- created a lot of noise.

NEW ACCURACY

```
Average Precision Recall      F1    Per Label
negative          0.199      0.500      0.282
positive          0.833      0.555      0.666

Macro Average Precision Recall      F1    Over All Labels
            0.516      0.527      0.474

Label Counts {'negative': 2236, 'positive': 2678}
Micro Average Precision Recall      F1    Over All Labels
            0.545      0.530      0.491
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<435>100 |
negative | 366 <99>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.534
```

MovieReviews_3.py -- Adding sentiment detection

To satisfy the requirement of step 3 part B, we implemented sentiment math -- **sentiMaths**, if you will -- functions to calculate the percentages of different sentiments within each document.

STARTING POINT ACCURACY

0.511 (see above)
0.534 **no neutrals** (see above)

WHAT WAS IMPLEMENTED

Preliminary Sentiment analysis, utilizing **subjclueslen1-HLTEMNLP05.tff**

CODE

```
def readSubjectivity(path):
  flexicon = open(path, 'r')
  sldict = { }
  for line in flexicon:
    fields = line.split()
    strength = fields[0].split("=")[1]
    word = fields[2].split("=")[1]
    posTag = fields[3].split("=")[1]
    stemmed = fields[4].split("=")[1]
    polarity = fields[5].split("=")[1]
    if (stemmed == 'y'):
      isStemmed = True
    else:
      isStemmed = False
    sldict[word] = [strength, posTag, isStemmed, polarity]
  return sldict

SLpath = "./SentimentLexicons/subjclueslen1-HLTEMNLP05.tff"
SL = readSubjectivity(SLpath)

negationwords = ['no', 'not', 'never', 'none', 'nowhere', 'nothing',
'noone', 'rather', 'hardly', 'scarcely', 'rarely', 'seldom', 'neither',
'nor']
def generateFeatureSets(document, SL, negationwords):
  document_words = set(document)
  # print('LENGTH', len(document_words))
  features = {}
  features['length'] = len(document_words)
  weakPos = 0
  strongPos = 0
  weakNeg = 0
  strongNeg = 0
```

```
negationWords = 0
psc = 0
nsc = 0
for word in document_words:
  if word in negationwords:
    negationWords +=1
  features['negationwords'] = negationWords
  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
    psc = (weakPos) + (strongPos)
    nsc = (weakNeg) + (strongNeg)
    features['positiveStrengthCount'] = (2 * weakPos) + (5 * strongPos)
    features['negativeStrengthCount'] = (2 * weakNeg) + (5 * strongNeg)
length = len(document_words)
if length > 10:
  features['percpositive'] = round(psc/length*100,2)
  features['percnegative'] = round(nsc/length*100,2)
print(features)
return features
```

IN ENGLISH

Added a function that utilized an external human-made document categorizing words as strong/weak positive/negative. Stored these values as well as percentages in our features object.

NEW ACCURACY

```
Average Precision Recall      F1    Per Label
negative          0.485      0.445      0.464
neutral           0.631      0.716      0.671
positive          0.586      0.510      0.545

Macro Average Precision Recall      F1    Over All Labels
            0.568      0.557      0.560
```

```
Label Counts {'negative': 2236, 'neutral': 5086, 'positive': 2678}
Micro Average Precision Recall      F1    Over All Labels
            0.587      0.600      0.591
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<310> 88 106 |
positive |  53<148> 67 |
negative |  73  47<108>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.566
```

MovieReviews_3b.py -- Adding sentiment detection & Removing Neutrals

STARTING POINT ACCURACY
0.566

WHAT WAS IMPLEMENTED
Removed neutrals

CODE

```
  phrasedocs = []
  neutraldocs = []
  for phrase in phraselist:
    tokens = nltk.word_tokenize(phrase[0])
    sentiment = int(phrase[1])
    if (sentiment == 2):
      # phrasedocs.append((tokens, 'neutral'))
      neutraldocs.append((tokens, 'neutral'))
    if ((sentiment == 0) or (sentiment == 1)):
      phrasedocs.append((tokens, 'negative'))
    if ((sentiment == 3) or (sentiment == 4)):
      phrasedocs.append((tokens, 'positive'))
```

IN ENGLISH

Since we only want to see if we can predict negative or positive, neutrals -- often partial phrases or single words, in this particular dataset -- created a lot of noise.

NEW ACCURACY

```
Average Precision Recall      F1    Per Label
negative          0.591      0.698      0.639
positive          0.785      0.697      0.738

Macro Average Precision Recall      F1    Over All Labels
            0.688      0.697      0.689

Label Counts {'negative': 2236, 'positive': 2678}
Micro Average Precision Recall      F1    Over All Labels
            0.697      0.697      0.693
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<395>140 |
negative | 175<290>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.685
```

MovieReviews_4.py -- Adding BOW sparse matrix

STARTING POINT ACCURACY

0.566 (see above)
0.685 **no neutrals** (see above)

WHAT WAS IMPLEMENTED

Frequency Distributions

CODE

```
  all_words_list = [word for (sent,cat) in docs for word in sent]
  all_words = nltk.FreqDist(all_words_list)
  word_items = all_words.most_common(1500)
  word_features = [word for (word,count) in word_items]

  featuresets = [(generateFeatureSets(d, SL, word_features), c) for (d, c)
in docs]
```

IN ENGLISH

Created a sparse matrix of the most common words

NEW ACCURACY

```
Average Precision Recall      F1    Per Label
negative          0.489      0.509      0.498
positive          0.528      0.617      0.569
neutral           0.739      0.676      0.706

Macro Average Precision Recall      F1    Over All Labels
            0.585      0.601      0.591

Label Counts {'negative': 2236, 'positive': 2678, 'neutral': 5086}
Micro Average Precision Recall      F1    Over All Labels
            0.627      0.623      0.623
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<366> 53  85 |
positive |  85<139> 44 |
negative |  94  24<110>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.615
```

MovieReviews_4b.py -- Adding BOW sparse matrix & Removing Neutrals

STARTING POINT ACCURACY

0.615

WHAT WAS IMPLEMENTED

Removed neutrals

CODE

```
phrasedocs = []
neutraldocs = []
for phrase in phraselist:
  tokens = nltk.word_tokenize(phrase[0])
  sentiment = int(phrase[1])
  if (sentiment == 2):
    # phrasedocs.append((tokens, 'neutral'))
    neutraldocs.append((tokens, 'neutral'))
  if ((sentiment == 0) or (sentiment == 1)):
    phrasedocs.append((tokens, 'negative'))
  if ((sentiment == 3) or (sentiment == 4)):
    phrasedocs.append((tokens, 'positive'))
```

IN ENGLISH

Since we only want to see if we can predict negative or positive, neutrals -- often partial phrases or single words, in this particular dataset -- created a lot of noise.

NEW ACCURACY

```
Average Precision Recall      F1    Per Label
negative          0.723      0.775      0.748
positive          0.825      0.781      0.802

Macro Average Precision Recall      F1    Over All Labels
            0.774      0.778      0.775

Label Counts {'negative': 2236, 'positive': 2678}
Micro Average Precision Recall      F1    Over All Labels
            0.779      0.779      0.778
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<423>112 |
negative | 114<351>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.774
```

MovieReviews_5.py -- Removing Stopwords

STARTING POINT ACCURACY

0.545 (see above)
0.698 **no neutrals** (see above)

WHAT WAS IMPLEMENTED

Removed Stopwords

CODE

```
  stop_words = set(stopwords.words('english'))
  all_words_list = [word for (sent,cat) in docs for word in sent]
  all_words_list_ns = [word for (sent,cat) in docs for word in sent if not
word in stop_words]
  print(len(all_words_list_ns))

  all_words = nltk.FreqDist(all_words_list)
  all_words_ns = nltk.FreqDist(all_words_list_ns)

  word_items = all_words.most_common(2000)
  word_items_ns = all_words_ns.most_common(2000)
  word_features = [word for (word,count) in word_items]
  word_features_ns = [word for (word,count) in word_items_ns]

  featuresets = [(generateFeatureSets(d, SL, negationwords,
word_features_ns), c) for (d, c) in docs]
```

IN ENGLISH

Removed stopwords

NEW ACCURACY

```
Average Precision Recall      F1    Per Label
neutral           0.720      0.691      0.705
negative          0.511      0.538      0.524
positive          0.580      0.603      0.592

Macro Average Precision Recall      F1    Over All Labels
            0.604      0.611      0.607

Label Counts {'neutral': 5086, 'negative': 2236, 'positive': 2678}
Micro Average Precision Recall      F1    Over All Labels
            0.636      0.633      0.634
```

```
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<348> 70  86 |
positive |  72<162> 34 |
negative |  78  28<122>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.632
```

MovieReviews_5b.py -- Removing Stopwords & Removing Neutrals

STARTING POINT ACCURACY
0.632

WHAT WAS IMPLEMENTED

Removed neutrals

CODE

```
  phrasedocs = []
  neutraldocs = []
  for phrase in phraselist:
    tokens = nltk.word_tokenize(phrase[0])
    sentiment = int(phrase[1])
    if (sentiment == 2):
      # phrasedocs.append((tokens, 'neutral'))
      neutraldocs.append((tokens, 'neutral'))
    if ((sentiment == 0) or (sentiment == 1)):
      phrasedocs.append((tokens, 'negative'))
    if ((sentiment == 3) or (sentiment == 4)):
      phrasedocs.append((tokens, 'positive'))
```

IN ENGLISH

Since we only want to see if we can predict negative or positive, neutrals -- often partial phrases or single words, in this particular dataset -- created a lot of noise.

NEW ACCURACY -- ACCURACY DROPPED

```
Average Precision Recall      F1    Per Label
negative          0.729      0.793      0.759
positive          0.841      0.788      0.813

Macro Average Precision Recall      F1    Over All Labels
            0.785      0.791      0.786

Label Counts {'negative': 2236, 'positive': 2678}
Micro Average Precision Recall      F1    Over All Labels
            0.790      0.790      0.789
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<434>101 |
negative | 114<351>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.785
```

MovieReviews_6.py -- Bigrams

STARTING POINT ACCURACY

0.632 (see above)
0.785 **no neutrals** (see above)

WHAT WAS IMPLEMENTED

Added bigrams

CODE

```
 finder = BigramCollocationFinder.from_words(all_words_list)
 bigram_features = finder.nbest(bigram_measures.pmi, 500)

 featuresets = [(generateFeatureSets(d, SL, negationwords,
word_features_ns, bigram_features), c) for (d, c) in docs]
```

IN ENGLISH

Added bigrams

NEW ACCURACY

```
Average Precision Recall      F1    Per Label
neutral          0.720      0.691      0.705
negative         0.511      0.538      0.524
positive         0.580      0.603      0.591

Macro Average Precision Recall      F1    Over All Labels
            0.604      0.611      0.607

Label Counts {'neutral': 5086, 'negative': 2236, 'positive': 2678}
Micro Average Precision Recall      F1    Over All Labels
            0.636      0.633      0.634
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<348> 70  86 |
positive |  72<162> 34 |
negative |  78  28<122>|
---------+-------------+
(row = reference; col = test)

Overall Accuracy 0.632
```

Unfortunately, literally nothing changed

MovieReviews_6b.py -- Bigrams & Removing Neutrals

STARTING POINT ACCURACY
0.632

WHAT WAS IMPLEMENTED
Removed neutrals

CODE

```
  phrasedocs = []
  neutraldocs = []
  for phrase in phraselist:
    tokens = nltk.word_tokenize(phrase[0])
    sentiment = int(phrase[1])
    if (sentiment == 2):
      # phrasedocs.append((tokens, 'neutral'))
      neutraldocs.append((tokens, 'neutral'))
    if ((sentiment == 0) or (sentiment == 1)):
      phrasedocs.append((tokens, 'negative'))
    if ((sentiment == 3) or (sentiment == 4)):
      phrasedocs.append((tokens, 'positive'))
```

IN ENGLISH

Since we only want to see if we can predict negative or positive, neutrals -- often partial phrases or single words, in this particular dataset -- created a lot of noise.

NEW ACCURACY

```
Average Precision Recall      F1    Per Label
negative          0.729      0.793      0.759
positive          0.841      0.788      0.813

Macro Average Precision Recall      F1    Over All Labels
            0.785      0.791      0.786

Label Counts {'negative': 2236, 'positive': 2678}
Micro Average Precision Recall      F1    Over All Labels
            0.790      0.790      0.789
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<434>101 |
negative | 114<351>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.785
```

Unfortunately, literally nothing changed again.

MovieReviews_7.py -- POS

STARTING POINT ACCURACY

0.632 (see above)
0.785 **no neutrals** (see above)

WHAT WAS IMPLEMENTED

Added POS count via nltk.pos_tag

CODE

```
 finder = BigramCollocationFinder.from_words(all_words_list)
 bigram_features = finder.nbest(bigram_measures.pmi, 500)

 featuresets = [(generateFeatureSets(d, SL, negationwords,
word_features_ns, bigram_features), c) for (d, c) in docs]
```

IN ENGLISH

Tallied the number of different parts of speech

NEW ACCURACY -- Accuracy went down

```
46044

Average Precision Recall      F1    Per Label
positive          0.549      0.613      0.579
neutral           0.740      0.684      0.711
negative          0.503      0.537      0.519

Macro Average Precision Recall      F1    Over All Labels
            0.597      0.611      0.603

Label Counts {'positive': 2678, 'neutral': 5086, 'negative': 2236}
Micro Average Precision Recall      F1    Over All Labels
            0.636      0.632      0.633
         |       p   n |
         |   n   o   e |
         |   e   s   g |
         |   u   i   a |
         |   t   t   t |
         |   r   i   i |
         |   a   v   v |
         |   l   e   e |
---------+-------------+
 neutral |<357> 61  86 |
positive |  82<150> 36 |
negative |  86  25<117>|
---------+-------------+
(row = reference; col = test)
```

```
Overall Accuracy 0.624
```

Unfortunately, overall accuracy decreased.

MovieReviews_7b.py -- POS & Removing Neutrals

STARTING POINT ACCURACY
0.632 (best) 0.624 (previous)

WHAT WAS IMPLEMENTED

Removed neutrals

CODE

```
phrasedocs = []
neutraldocs = []
for phrase in phraselist:
  tokens = nltk.word_tokenize(phrase[0])
  sentiment = int(phrase[1])
  if (sentiment == 2):
    # phrasedocs.append((tokens, 'neutral'))
    neutraldocs.append((tokens, 'neutral'))
  if ((sentiment == 0) or (sentiment == 1)):
    phrasedocs.append((tokens, 'negative'))
  if ((sentiment == 3) or (sentiment == 4)):
    phrasedocs.append((tokens, 'positive'))
```

IN ENGLISH

Since we only want to see if we can predict negative or positive, neutrals -- often partial phrases or single words, in this particular dataset -- created a lot of noise.

NEW ACCURACY

```
Average Precision Recall      F1    Per Label
positive          0.843      0.789      0.815
negative          0.729      0.795      0.761
```

```

Macro Average Precision Recall      F1    Over All Labels
            0.786      0.792      0.788

Label Counts {'positive': 2678, 'negative': 2236}
Micro Average Precision Recall      F1    Over All Labels
            0.791      0.792      0.790
         |   p   n |
         |   o   e |
         |   s   g |
         |   i   a |
         |   t   t |
         |   i   i |
         |   v   v |
         |   e   e |
---------+---------+
positive |<435>100 |
negative | 113<352>|
---------+---------+
(row = reference; col = test)

Overall Accuracy 0.787
```

Further study on Experiments Part Two is ongoing!
*For a sample of 500, with neutral removed :D