CSI4107 Assignment 2 Report

Shaughn Finnerty (6300433)

Part 1

In the first part of the experiment, Python was used to read the twitter messages from the text file. Using Scikit-learn, a machine learning toolkit for Python, we are able to create a $n * m$ matrix for n documents with m features of words using a `CountVectorizer` object.

The `CountVectorizer` object takes an array of text objects representing documents and creates an appropriate matrix representing the counts of token words for each document. Documents are first preprocessed with a preprocess object, and then tokenized with a tokenizer object. Together, these form an analyzer that is called to process every document. We decided to extend the basic analyzer by stemming all the words produced by the preprocessor and tokenizer using the `EnglishStemmer` provided by Natural Language Toolkit (NLTK).

Using the matrix created from this preprocessing, tokenization, and stemming, a sparse arff is then able to be produced for use in Weka. In the sparse arff file, a twitter document is represented by the index of the token in the bag of words list and the count of that token in that document. Tokens are only specified if they are present in the document. This reduces arff file size as features (i.e. words) not present are not included and it is implied that they are 0 for a given document.

With this arff file, the first run in Weka resulted in the following results from a 10-fold cross validation with the three different classifiers:

Decision Tree:

```
=== Stratified cross-validation ===

Correctly Classified Instances        3455               47.7936 %
Incorrectly Classified Instances      3774               52.2064 %
Kappa statistic                          0.2297
Mean absolute error                      0.28
Root mean squared error                  0.4545
Relative absolute error                 80.8692 %
Root relative squared error            109.2265 %
Total Number of Instances             7229


=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.692     0.365      0.612     0.692     0.649      0.704    positiv
e
                 0.354     0.127      0.379     0.354     0.366      0.636    negativ
e
                 0.224     0.155      0.282     0.224     0.249      0.556    neutral
                 0.344     0.115      0.351     0.344     0.348      0.641    objecti
ve
Weighted Avg.    0.478     0.239      0.46      0.478     0.467      0.651


=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 2271  363  387  263 |    a = positive
  486  458  214  135 |    b = negative
  634  263  346  304 |    c = neutral
  319  125  281  380 |    d = objective
```

Naive Bayes:

```
=== Stratified cross-validation ===

Correctly Classified Instances        3368               46.5901 %
Incorrectly Classified Instances      3861               53.4099 %
Kappa statistic                          0.244
Mean absolute error                      0.2824
Root mean squared error                  0.445
Relative absolute error                 81.5583 %
Root relative squared error            106.9465 %
Total Number of Instances             7229


=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.582     0.279      0.635     0.582     0.607      0.705    positiv
e
                 0.452     0.183      0.35      0.452     0.395      0.698    negativ
e
                 0.219     0.13       0.315     0.219     0.258      0.597    neutral
                 0.482     0.153      0.363     0.482     0.414      0.745    objecti
ve
Weighted Avg.    0.466     0.211      0.474     0.466     0.465      0.687


=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 1911  596  363  414 |    a = positive
  369  585  184  155 |    b = negative
  480  361  339  367 |    c = neutral
  251  130  191  533 |    d = objective
```

Support Vector Machine (SMO):

```
=== Stratified cross-validation ===

Correctly Classified Instances        3698               51.1551 %
Incorrectly Classified Instances      3531               48.8449 %
Kappa statistic                          0.2741
Mean absolute error                      0.3202
Root mean squared error                  0.4063
Relative absolute error                 92.476  %
Root relative squared error             97.6539 %
Total Number of Instances             7229


=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.725     0.36       0.626     0.725     0.672      0.716    positiv
e
                 0.371     0.103      0.439     0.371     0.402      0.718    negativ
e
                 0.299     0.162      0.334     0.299     0.316      0.579    neutral
                 0.338     0.094      0.394     0.338     0.364      0.718    objecti
ve
Weighted Avg.    0.512     0.231      0.495     0.512     0.5        0.688



=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 2382  291  396  215 |    a = positive
  462  480  253   98 |    b = negative
  601  223  463  260 |    c = neutral
  359  100  273  373 |    d = objective
```

Clearly, the SVM classifier produced the best results with 51.15% correctly classified instances and a precision of 49.5%.

Part 2

Emoticons, Question Marks, Exclamation Marks

When adding features to the bag of words feature set, first begin by counting the amount of smiley-based emoticons and sad-based emoticons. This analysis was carried out on each document using the following code:

```
additional_features["smilies"] = twitter_document.msg_text.count("(:") + twitter_document.msg_text.count(":)") + twitter_document.msg_text.count(":-)") + twitter_document.msg_text.count(":o)") + twitter_document.msg_text.count(":]") + twitter_document.msg_text.count(":3") + twitter_document.msg_text.count(":c)") + 2*twitter_document.msg_text.count(":D") + 2*twitter_document.msg_text.count("C:")
additional_features["exclamations"] = twitter_document.msg_text.count("!")
additional_features["questions"] = twitter_document.msg_text.count("?")
additional_features["sadfaces"] = twitter_document.msg_text.count("):") + twitter_document.msg_text.count(":(") + twitter_document.msg_text.count(":-(") + twitter_document.msg_text.count(":c") + twitter_document.msg_text.count(":[") + 2*twitter_document.msg_text.count("D8") + twitter_document.msg_text.count("D;") + 2*twitter_document.msg_text.count("D=") + twitter_document.msg_text.count("DX");
```

The following emoticons representing smilies were seached for:
`(: , :) , :-) , o) , :] , :3 , :c , :D, C:`

The following emoticons representing sad faces were searched for:
`): , :( , :-( , :c , :[ , D8 , D; , D=, DX`

In addition, the amount of question marks and exclamations were added to each document as features.

This resulted in the following results from the three classifiers:

Decision Tree:

```
=== Stratified cross-validation ===

Correctly Classified Instances        3578               49.4951 %
Incorrectly Classified Instances      3651               50.5049 %
Kappa statistic                          0.254
Mean absolute error                      0.2737
Root mean squared error                  0.4489
Relative absolute error                 79.036  %
Root relative squared error            107.8775 %
Total Number of Instances             7229


=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.716     0.346      0.633     0.716     0.672      0.721    positiv
e
                 0.351     0.13       0.371     0.351     0.361      0.614    negativ
e
                 0.262     0.154      0.316     0.262     0.286      0.572    neutral
                 0.333     0.105      0.364     0.333     0.348      0.633    objecti
ve
Weighted Avg.    0.495     0.229      0.477     0.495     0.484      0.657


=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 2351  354  358  221 |    a = positive
  461  454  242  136 |    b = negative
  566  291  405  285 |    c = neutral
  336  125  276  368 |    d = objective
```

Naive Bayes:

```
=== Stratified cross-validation ===

Correctly Classified Instances        3459               47.8489 %
Incorrectly Classified Instances      3770               52.1511 %
Kappa statistic                          0.271
Mean absolute error                      0.2747
Root mean squared error                  0.443
Relative absolute error                 79.3219 %
Root relative squared error            106.4743 %
Total Number of Instances             7229


=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.581     0.224      0.684     0.581     0.628      0.73     positiv
e
                 0.5       0.198      0.355     0.5       0.415      0.709    negativ
e
                 0.218     0.124      0.325     0.218     0.261      0.605    neutral
                 0.512     0.165      0.359     0.512     0.422      0.753    objecti
ve
Weighted Avg.    0.478     0.189      0.499     0.478     0.48       0.703


=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 1909  613  334  428 |    a = positive
  303  646  174  170 |    b = negative
  393  404  338  412 |    c = neutral
  186  159  194  566 |    d = objective
```

SVM:

```
=== Stratified cross-validation ===

Correctly Classified Instances        3773               52.1926 %
Incorrectly Classified Instances      3456               47.8074 %
Kappa statistic                          0.2935
Mean absolute error                      0.3183
Root mean squared error                  0.4041
Relative absolute error                 91.9333 %
Root relative squared error             97.1217 %
Total Number of Instances             7229


=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.732     0.33       0.649     0.732     0.688      0.736    positiv
e
                 0.394     0.107      0.446     0.394     0.419      0.724    negativ
e
                 0.305     0.165      0.335     0.305     0.32       0.586    neutral
                 0.351     0.096      0.398     0.351     0.373      0.724    objecti
ve
Weighted Avg.    0.522     0.219      0.507     0.522     0.513      0.7


=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 2403  282  391  208 |    a = positive
  426  510  250  107 |    b = negative
  555  249  472  271 |    c = neutral
  321  102  294  388 |    d = objective
```

As you can see this increased the average precision for all classifiers. Most notably, the SVM classifier increased from **49.5% to 50.7%.** This classifier continued to be be the most accurate, correctly classifying **3773** twitter messages or 52.2%.

SentiWordNet for Positive, Negative, Objective Scores

In trying to continue the improvement of the classifiers, we used senti wordnet to add positive, negative, and objective scores for each document. Iterating through each document, each word was analyzed using senti wordnet and the positive, negative, and objective score for the word (in all of the synsets in which it belongs) was added to to total positive, negative and objective score for the document. This was achieved using the following code:

```
for word in twitter_document.msg_text.split():
    for synset in swn.senti_synsets(word):
        additional_features["posscore"] += synset.pos_score()
        additional_features["negscore"] += synset.neg_score()
        additional_features["objscore"] += synset.obj_score()
```

3 features were added to the arff file: `posscore, negscore, objscore`

The three classifiers then provided the following results with these new features:

Decision Tree:

```
=== Stratified cross-validation ===

Correctly Classified Instances        3613               49.9793 %
Incorrectly Classified Instances      3616               50.0207 %
Kappa statistic                          0.2636
Mean absolute error                      0.2698
Root mean squared error                  0.4552
Relative absolute error                 77.9164 %
Root relative squared error            109.3981 %
Total Number of Instances             7229


=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.715     0.332      0.642     0.715     0.676      0.714    positiv
e
                 0.364     0.125      0.388     0.364     0.376      0.612    negativ
e
                 0.266     0.156      0.317     0.266     0.289      0.565    neutral
                 0.348     0.111      0.362     0.348     0.355      0.627    objecti
ve
Weighted Avg.    0.5       0.224      0.484     0.5       0.49       0.651


=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 2347  344  356  237 |    a = positive
  442  471  242  138 |    b = negative
  557  277  411  302 |    c = neutral
  312  123  286  384 |    d = objective
```

Naive Bayes:

```
=== Stratified cross-validation ===

Correctly Classified Instances        3419               47.2956 %
Incorrectly Classified Instances      3810               52.7044 %
Kappa statistic                          0.2739
Mean absolute error                      0.2728
Root mean squared error                  0.4476
Relative absolute error                 78.7973 %
Root relative squared error            107.5843 %
Total Number of Instances             7229


=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.548     0.183      0.714     0.548     0.62        0.736    positiv
e
                 0.491     0.191      0.36      0.491     0.415       0.722    negativ
e
                 0.239     0.136      0.323     0.239     0.275       0.601    neutral
                 0.555     0.193      0.342     0.555     0.423       0.757    objecti
ve
Weighted Avg.    0.473     0.176      0.51      0.473     0.479       0.708


=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 1801  606  362  515 |    a = positive
  242  635  210  206 |    b = negative
  326  390  370  461 |    c = neutral
  155  135  202  613 |    d = objective
```

SVM:

```
=== Stratified cross-validation ===

Correctly Classified Instances        3792               52.4554 %
Incorrectly Classified Instances      3437               47.5446 %
Kappa statistic                          0.2979
Mean absolute error                      0.3175
Root mean squared error                  0.4031
Relative absolute error                 91.7069 %
Root relative squared error             96.8872 %
Total Number of Instances             7229


=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.731     0.328      0.65      0.731     0.688      0.738    positiv
e
                 0.398     0.103      0.456     0.398     0.425      0.73     negativ
e
                 0.31      0.164      0.34      0.31      0.324      0.588    neutral
                 0.361     0.098      0.4       0.361     0.38       0.728    objecti
ve
Weighted Avg.    0.525     0.218      0.511     0.525     0.516      0.703


=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 2400  274  392  218 |    a = positive
  426  514  249  104 |    b = negative
  550  242  479  276 |    c = neutral
  318   98  290  399 |    d = objective
```

Again, we see an increase in precision and correctly classified instances for all classifiers. Most notably, the SVM classifier increased from **50.7% to 51.1%.** This classifier continued to be be the most accurate, correctly classifying **3792** twitter messages or 52.45%.

With these results it was noticed that combining bag of words with counting exclamations, question marks, smile emoticons, sad emoticons, and analyzing the sentiment of each individual word in a Twitter document can in fact increase precision for classifiers.

In addition the precision by each class is more balanced with the SVM classifier. This is likely more favorable since the average precision isn't increasing from a class that is constantly being assigned (e.g. positive) and thus resulting in a false precision value. We see an increase in precision for all classes using this classifier, indicating that it is a genuine increase in average precision.

Feature Selection with Chi^2

The final alteration that proved to increase precision and correctly classified instances significantly was using feature selection techniques on the word features (i.e. reducing the dimensionality of the feature vectors by selecting the features that are most representative).

Scikit-learn provides various feature selection tools. The one that was used for this experiment was the `SelectKBest` class. This class removes all but the `k` highest scoring features when analyzed using a specific scoring function (in our experiment, we used the `chi2` scoring function. The analysis is based on univariate statistical tests that determines the "usefulness" of the feature for classification and this part is often considered part of the preprocessing. The dimensionality was reduced to 2000 word features using the following Python code:

```
#Note that self.k = 2000
self.feature_matrix_token_counts = SelectKBest(chi2,self.k).fit_transform(self.feature_matri
x_token_counts, all_twitter_msg_polarity)
self.token_feature_names = [i for i in range(self.feature_matrix_token_counts.shape[1])]
self.amount_of_token_features = self.feature_matrix_token_counts.shape[1]
```

Unfortunately the names of the features (i.e. the words) could not be preserved in this matrix so they were renamed just by their index in the feature vector. Additional features (i.e. emoticons, positive, negative, objective scores, exclamation counts, and question mark counts) were added to support these newly chosen tokens (as they had previously shown to improve the other experiments). The final results for these classifiers was much better than before:

Best Results:

Decision Tree:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3542               48.9971 %
Incorrectly Classified Instances      3687               51.0029 %
Kappa statistic                          0.244
Mean absolute error                      0.2747
Root mean squared error                  0.4528
Relative absolute error                 79.3429 %
Root relative squared error            108.823  %
Total Number of Instances             7229

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.718     0.36       0.624     0.718     0.668      0.713    positiv
e
                 0.353     0.123      0.384     0.353     0.368      0.623    negativ
e
                 0.239     0.156      0.294     0.239     0.263      0.552    neutral
                 0.324     0.106      0.355     0.324     0.339      0.63     objecti
ve
Weighted Avg.    0.49      0.235      0.469     0.49      0.477      0.65

=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 2359  327  361  237 |    a = positive
  454  456  254  129 |    b = negative
  626  268  369  284 |    c = neutral
  341  135  271  358 |    d = objective
```

Naive Bayes:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3476               48.0841 %
Incorrectly Classified Instances      3753               51.9159 %
Kappa statistic                          0.2838
Mean absolute error                      0.2705
Root mean squared error                  0.4433
Relative absolute error                 78.1104 %
Root relative squared error            106.5535 %
Total Number of Instances             7229

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.558     0.183      0.718     0.558     0.628      0.74     positiv
e
                 0.5       0.183      0.373     0.5       0.427      0.73     negativ
e
                 0.243     0.133      0.333     0.243     0.281      0.608    neutral
                 0.561     0.194      0.342     0.561     0.425      0.762    objecti
ve
Weighted Avg.    0.481     0.174      0.516     0.481     0.487      0.714

=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 1833  574  357  520 |    a = positive
  236  647  202  208 |    b = negative
  328  380  376  463 |    c = neutral
  157  133  195  620 |    d = objective
```

SMO:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4353               60.2158 %
Incorrectly Classified Instances      2876               39.7842 %
Kappa statistic                          0.4022
Mean absolute error                      0.3034
Root mean squared error                  0.3866
Relative absolute error                 87.622  %
Root relative squared error             92.9087 %
Total Number of Instances             7229

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.823     0.332      0.674     0.823     0.741      0.771    positiv
e
                 0.475     0.071      0.594     0.475     0.528      0.793    negativ
e
                 0.364     0.119      0.454     0.364     0.404      0.655    neutral
                 0.427     0.077      0.502     0.427     0.461      0.777    objecti
ve
Weighted Avg.    0.602     0.201      0.586     0.602     0.588      0.751

=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 2704  192  242  146 |    a = positive
  412  614  193   74 |    b = negative
  575  160  563  249 |    c = neutral
  323   68  242  472 |    d = objective
```

Clearly, using the feature selection tools to keep only the word features that are most useful is a huge step in helping the Naive Bayes and SVM classifier's accuracies.

Most notably, we see a large jump in the SVM classifier, jumping from an average precision of 51.1 to **58.5%**. In addition the precision for each class increases so there is less of a difference between each class. Thus, there is no single class dominating and providing a falsely high average precision. In addition the correctly classified messages jumps from 3792 (52.45%) to **4353** or **60.22%**.

Other Features Explored

n-grams

After reading some papers exploring sentiment analysis, it was seen that using n-grams as features in a bag of words approach can sometimes improve accuracy especially for corpus in which words can be frequently mispelled. We used n-grams created within word boundaries so as to allow for mispellings to occur. In order to do this Scikit-learn's CountVectorizer options for creating features of n-grams were used. The CountVectorizer to do this was created with this command:

```
vectorizer = CountVectorizer(stop_words='english', min_df=2, analyzer="char_wb", ngram_range
=(1,2))
```

As a result, unigrams and bi-grams were created as features and their counts were recorded throughout the corpus for each document. *We also added the additional features that were shown to improve precision in the previous results shown.*

We did not perform **SelectKBest** on these n-grams.

The initial test to see if this would increase precision was using the Naive Bayes classifier with this new data representation for the documents. The stratified cross-validation results became:

```
=== Stratified cross-validation ===

Correctly Classified Instances        3087               42.703  %
Incorrectly Classified Instances      4142               57.297  %
Kappa statistic                          0.2207
Mean absolute error                      0.2879
Root mean squared error                  0.5093
Relative absolute error                 83.1383 %
Root relative squared error            122.4172 %
Total Number of Instances             7229


=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.479     0.197      0.669     0.479     0.558      0.682    positiv
e
                 0.567     0.265      0.318     0.567     0.407      0.683    negativ
e
                 0.112     0.083      0.27      0.112     0.159      0.57     neutral
                 0.548     0.215      0.315     0.548     0.4        0.723    objecti
ve
Weighted Avg.    0.427     0.188      0.467     0.427     0.422      0.664


=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 1574  841  259  610 |    a = positive
  235  733  102  223 |    b = negative
  372  515  174  486 |    c = neutral
  172  217  110  606 |    d = objective
```

The average precision for this classifier dropped from 51.0% to 46.7% (when compared to the best results retrieved with the Naive Bayes without SelectKBest) and the correctly classified instances dropped from 47% to 42%. With these initial results, it was decided to not continue with unigrams and bigrams as features. While the reason for this drop is unknown it is possible that since the unigrams and bigrams were made out of characters (in an attempt to disambiguate mispellings), that the usefulness that words on their own provided was lost.

Normalizing Positive, Negative, Objective Scores

One thing that was noticed when analyzing the positive, negative, and objective scores with Senti Wordnet was that some values, particularly objective score were very high. It was decided to see if normalization would make a difference. This was done by finding the maximum positive, negative, and objective score in the corpus and dividing all other documents' scores by the respective maximum score.

This did not impact the accuracy of the classifiers positively. The decision tree average precision dropped by almost 1%. The Naive Bayes classifier average precission did not change. And the SVM classifier dropped in average precision by 0.3%.

The results for these classifiers with this data representation via cross-fold validation can be found here.

Tf-Idf Weights

Similar results were seen when using Tf-Idf weights for the frequency of a word/token in the bag of words feature matrix. The Tf-Idf representation was simply achieved using the Scikit-learn `TfidfTransformer` object.

It was created using the following python code:

```
transformer = TfidfTransformer()
tf_idf_feature_matrix = transform.fit_transform(self.feature_matrix_token_counts)
```

This tf-idf feature matrix was then used to create a sparse arff file with all of the additional features that contributed to increased precision and it was evaluated in Weka using the Naive Bayes classifier. Due to the initial results from this classifier (decreased precision and correctly classified instances), it was decided not to continue with the lengthier SVM and Decision Tree classifiers.

```
=== Stratified cross-validation ===

Correctly Classified Instances        2869               39.6874 %
Incorrectly Classified Instances      4360               60.3126 %
Kappa statistic                          0.2018
Mean absolute error                      0.3015
Root mean squared error                  0.5468
Relative absolute error                 87.0684 %
Root relative squared error            131.4066 %
Total Number of Instances             7229


=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.389     0.136      0.705     0.389     0.501      0.694    positiv
e
                 0.65      0.372      0.276     0.65      0.387      0.692    negativ
e
                 0.109     0.074      0.286     0.109     0.157      0.563    neutral
                 0.527     0.195      0.328     0.527     0.404      0.726    objecti
ve
Weighted Avg.    0.397     0.174      0.481     0.397     0.392      0.671


=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 1278 1223  225  558 |    a = positive
  177  841  105  170 |    b = negative
  243  670  168  466 |    c = neutral
  116  318   89  582 |    d = objective
```

Since the twitter messages are relatively the same length, the tf-idf normalizing weight should not be too crucial for this experiment.

Removing URLs, Hashtags, Usernames

The next venture was to experiment with replacing URLs, hashtags, usernames with simply "url, hashtag, userz" respectively. This was done using simple regex substitution that can be found in tools.py. Substitution was performed before the text was analyzed using the `CountVectorizer` so that usernames, urls, or hashtags appearing would only be the feature that is counted and different values for them would not be present as features. However, for the Naive Bayes classifier and the SMO classifier, the average precision and correctly classified instances dropped. Here is an example of the stratified cross-fold validation using SMO on this new data representation.

```
=== Stratified cross-validation ===

Correctly Classified Instances        3669               50.7539 %
Incorrectly Classified Instances      3560               49.2461 %
Kappa statistic                          0.2673
Mean absolute error                      0.3203
Root mean squared error                  0.4066
Relative absolute error                 92.5093 %
Root relative squared error             97.7135 %
Total Number of Instances             7229


=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.723     0.368      0.621     0.723     0.668      0.713    positiv
e
                 0.376     0.106      0.437     0.376     0.404      0.718    negativ
e
                 0.282     0.162      0.322     0.282     0.3        0.577    neutral
                 0.337     0.092      0.398     0.337     0.365      0.715    objecti
ve
Weighted Avg.    0.508     0.235      0.49      0.508     0.496      0.685


=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 2375  297  395  217 |    a = positive
  488  486  240   79 |    b = negative
  608  237  436  266 |    c = neutral
  356   93  284  372 |    d = objective
```

When 50.8% average precision was compared to an average 51.1% precision without the `SelectKBest` step it was deemed that the values of usernames, hashtags, and URLs sometimes provide relevant information for classifying twitter sentiment and so should be kept during the bag of words tokenization step. In addition the amount of correctly classified instances dropped by almost 2% without this information.

Conclusions & Notes

The results (predictions) from the best experiment using the counting of question marks, exclamations, emoticons, and positive, negative, and objective scores with the SVM classifier can be found in results.txt.

The four sentiment classes being present in the experiment certainly proved to be difficult as the confusion matrices show. Often, messages belong neutral and objective classes were confused with each other (and sometimes positive). It would have been beneficial to the accuracy of the system to possibly merge these classes or remove some of these messages but due to the requirements of the assignment, they were kept in throughout the entire experiment and a best attempt was made to increase the accuracy of the classifiers for all four classes.

The SVM results with SelectKBest features and additional analysis of emoticons, word scores, and punctuation brought a reasonable accuracy of **60.22%** correctly classified instances in a 10-fold cross validation.

All the while, other strategies (e.g. n-grams) were implemented and explored to determine the effect on results.

Dependencies

- Scikit-learn
- NLTK
- Both of these require the basic python scientific libraies:
  - numpy
  - scipy

- Once installed, NLTK SentiWordNet and WordNet data must be installed.

  - From a python interpreter run the commands:

  ```
  import nltk
  nltk.download()
  ```

  Go to all packages in the window prompt that opens and download the packages identified as `wordnet` and `sentiwordnet`.

Running

In the `vectorization.py` file `run()` method, you may need to change the following code:

```
arff_file_save_path = '/Users/shaughnfinnerty/code/school/csi4107/a2/arff/2000best-features-sparse-emoticon-questionmarks-exclamations-posscore-negscore-objscore.arff'
```

To match a file of your choice in which you want to save the arff file. Currently the vectorizer creates the arff file with the features that lead to the best SVM classifier results as discussed. However, you can enable some boolean values to create arff files for some of the approaches mentioned that did not increase results by changing the `__init__` method of the Vectorizer class here:

```
# These are properties used to control the features tested that did not increase results
self.filter_url_hashtag_username = False;
self.filter_numbers = False;
self.uni_bi_gram = False;
```

Other than that, simply run `python vectorization.py` and your arff file will be created with the data representation that lead to the best classification results.