BT2101 Individual Assignment 2
Tong Tsz Hin (Tony)

Summary:

I first focused on building functions that could process the data accurately to build prediction models, then moved into testing different types of training data based on factors like length of reviews using 3 different classifier methods – SVM, Naive Bayes and lexicon approach. Finally, SVM approach gave the best results for me, giving accuracy results of approximately 63%. Next, I tried using ensemble models to combine classifiers, which gave marginally better results than SVM classifier.

Data Pre-processing:

Using my 2 functions "textprocess()" and "english()", I eliminated all the non-English tweets and processed all the remaining words, using tolower(), removeNumbers(), stripWhitespace(), removePunctuation() and stemDocument(). I then added a new column for my training data, which contained the string of the category it was in. For example, a finance review would have a tag "finance" in its 2nd column, which was to be used for classification later.

Below is an example for the finance category:

Review	Class
"What a cool app to use"	"Finance"
"I can finally start saving money"	"Finance"
"I am going to manage my savings well"	"Finance"
…	…

After repeating this for the 5 categories, I merged them together to form the combined training data to run Naive Bayes and SVM. The impact of the data processing causes non-words like emoticons and punctuations to be removed, and stemming causes words with same meaning to be bundled together, so they are counted together when building the same prediction model.

Sample Size:

I ensured the category for each data is balanced using equal sample sizes when I built my prediction model. For example, I used 100 reviews from each category (totaling 500), and then moving on to 2000 reviews each category (totaling 10,000) to build my SVM classifier. I did this based on understanding that heavily skewing the training data classes could reduce the accuracy of the classifier.

Review Length:

In general the longer the review, the better the results tend to be as there are more words that imply certain categories. My training data was obtained from longer reviews of 20 words or more, and I tested the accuracy using reviews of different lengths ranging from 5, 10, 15 to 0 (no length restrictions) with results as shown below.

Useful Data:

For each category, I only used the review column as I deemed all the other information to be irrelevant in differentiating between the 5 categories. For author, date and rating, these data provided little insight into which category the app is from as people can download a variety of apps over time, and every app tends to get all the possible ratings (1-5) once the number of reviews goes large.

For title, even though it contained unstructured data, but it was usually too short to contain insightful words, which was why for my final review data, I only used reviews that contained more than 20 words for training.

Different Learning Methods:

Lexicon Classifier:

I initially started with 5 lists of words that had a high chance of identifying each category, words such as “cloudy, rainy, sunny” for weather and “money, savings, wealth” for finance. Each word is given an equal score of 1 and based on term frequency.
For each review, the classifier would then determine its category based on the highest number of words. For example, if the review contained 3 words “cloudy, rainy, sunny”, then the weather category would have a score of 3.

The highest score is then determined to become the response variable. If a review had no words from all of the 5 categories, then it’s labeled as NA, so I can use another classifier on it. Here are the lists of words I used:

Education: study, studying, education, educate, educating, books, book, mug, mugging, learn, exam, grade, teach

Game: play, addictive, addict, multiplayer, fun, game, kill, score, high

Finance: money, wealth, cash, bank, cheque, loan, finance, earn, earnings, earning, saving, save, account, accounting, gold

Weather: sunny, cloudy, windy, rainy, accurate, hot, cold, snow, hail, fair, warm, temperature, climate, weather

Social: connect, network, friends, friend, pal, chat, talk, audio, party, meetup, meet, call, social, love, dating, partner, message

At the start, only filtering for reviews longer than 20 words, I could get results for 40-50% of the sample data (rest being NA) with an accuracy of between 62-68%. However, 1 problem that occurred was mostly with the “social” category. The social category was too “strong”, or having words that affected the other categories, causing a large error rate.

	real				
com	weather	finance	game	social	education
weather	451	21	3	15	5
finance	92	1028	20	28	15
game	89	39	1194	76	18
social	197	138	137	303	129
education	3	4	1	17	687

Precision (weather) = 0.91
Precision (finance) = 0.87
Precision (game) = 0.84
Precision (social) = 0.34
Precision (education) = 0.96

Recall (weather) = 0.54
Recall (finance) = 0.84
Recall (game) = 0.88
Recall (social) = 0.69
Recall (education) = 0.80

F-Measure (weather) = 0.67
F-Measure (finance) = 0.85
F-Measure (game) = 0.86
F-Measure (social) = 0.46
F-Measure (education) = 0.87

Weighted Average (Precision) = 0.784
Weighted Average (Recall) = 0.75

The precision for social is low at 0.34 as the classifier wrongly classifies a large number of non-social reviews into the social category. Hence, I removed some words from the social category like "love, friend" that I deemed to be not concise enough, and I added "text" after looking at the common lingo in some of the reviews. This increased the accuracy to 70-76%.

Social (new list): connect, network, pal, chat, talk, party, meetup, meet, call, social, text, dating, partner, message

Word Length:

When I reduced the word length from 20 to 10, the accuracy still remains in the same range of 70-76%, which was surprising as I expected the performance of the classifier to decrease with lesser words available. However, the number of results drops to approximately 30-40 % as there are more NAs than before.

Conclusion:

Hence, the lexicon approach is a good way to predict the categories if at least 1 word from the word list appears in the review, which increases in chance the longer the review is.

Naive Bayes Classifier:

Using 1500 reviews from each category, I initially discovered that the accuracy was low at 35-45% and most of the reviews were wrongly classified under "game" and "weather". I only used reviews more than 20 words, as I wanted the most data to build the training model.

The following shows a testing data set of 500 I used the classifier on, with an accuracy of 0.45. For my training sets I've always used data sets of 500 (containing 100 reviews from each category that were not used in training)

```
> table
           real
com         education finance game social weather
  education        19       4    4      6       2
  finance          14      40    4     19      13
  game             41      20   80     31       8
  social            3       2    0     11       2
  weather          23      34   12     33      75
```

I then tried to use more training data (3000 per category) and try Naive Bayes once again. This time, I got an accuracy of 0.5. There was an improvement, but this came at a cost of using twice the amount of training data as compared to SVM. I could have still used even more data since it was in abundance in this case, but I did not want to use more as I felt the results it gave back were not worth the additional processing to build the model, and that I might not have the luxury of data in other cases.

Conclusion:

I decided that Naive Bayes was the least effective for this scenario.

SVM Classifier:

Using 1500 reviews from each category as training data for my SVM classifier, I initially noticed that the results were significantly better than Naïve Bayes classifier, using testing data that was also longer than 20 words.

```
           real
com         education finance game social weather
  education        58       8    5     13       7
  finance          18      78    4     10       4
  game              7       2   80      5       2
  social           14       7    8     68      10
  weather           3       5    3      4      77
>
```

The accuracy of this model was 0.72.

Precision (weather) = 0.84
Precision (finance) = 0.68
Precision (game) = 0.83
Precision (social) = 0.64
Precision (education) = 0.64

Recall (weather) = 0.77
Recall (finance) = 0.78
Recall (game) = 0.80
Recall (social) = 0.68
Recall (education) = 0.58

F-Measure (weather) = 0.80
F-Measure (finance) = 0.73
F-Measure (game) = 0.81
F-Measure (social) = 0.66
F-Measure (education) = 0.61

Weighted Average (Precision) = 0.73
Weighted Average (Recall) = 0.72

Now, I reduced the length of words to 10 and ran the prediction model again, with an accuracy of 0.70.

```
           real
com         education finance game social weather
  education        63      15   10     11       8
  finance           6      67    3      5       5
  game              7       3   78     11       4
  social           17      12    7     70       7
  weather           7       3    2      3      76
>
> sum(table)
[1] 500
> sum(diag(table))
[1] 354
> sum(diag(table))/ sum(table)
[1] 0.708
>
```

When the word length is reduced to 5 words and more, the accuracy drops to 0.63. This is understandable as there is less data available to work with. After trying multiple iterations, the accuracy levels can differ quite heavily, ranging from 60-75%.

I tried 5 testing data iterations of 500 reviews (100 from each category taken at random) and tested for accuracy, without any length restraints.

On average, my results gave about 63% accuracy.

```
> table
           real
com         education finance game social weather
  education        72      17    9     25      24
  finance           7      60    2      7       8
  game              5       3   80     12       3
  social           15      18    8     53      12
  weather           1       2    1      3      53
>
> sum(table)
[1] 500
> sum(diag(table))
[1] 318
> sum(diag(table))/ sum(table)
[1] 0.636
```

Conclusion:

I found SVM to have the best overall performance in terms of accuracy as compared to the other learning methods.

Ensemble Model:

I tried building an ensemble model based on voting from the 3 classifiers to improve the classification. What I observed was that if the SVM and lexicon classifier both had the same result, then it was almost always correct. Hence, I weighted 40% for SVM Classifier, 40% for Lexicon and 20% for Naïve Bayes. I did this by assigning each category a score of 2 if produced by the SVM and Lexicon classifier, and 1 if the Naive Bayes Classifier produced it.

If the lexicon classifier gave NA (meaning that no similar words were found) then the SVM classifier takes precedence. If all 3 classifier gave different results the lexicon classifier takes precedence, followed by SVM classifier. If SVM and lexicon classifier had different responses, the Naive Bayes classifier will decide the final vote.

```
> print(combined_table)
    svm_vector lexicon_vector nb_vector all_test_data.1.500..2.
1      weather           <NA>   weather               education
2    education           game education               education
3    education           <NA> education               education
4    education           <NA> education               education
5    education      education education               education
6         game           <NA>      game               education
7       social           <NA>      game               education
8    education           <NA> education               education
9       social           <NA>      game               education
10   education      education education               education
11   education           <NA>   weather               education
12      social           <NA>      game               education
13   education           <NA>      game               education
14   education      education   weather               education
15        game           game      game               education
16   education           <NA>      game               education
17   education           <NA>      game               education
```

Based on testing data longer than 10 words, I tried my ensemble model. Comparing the final results to the best results from the SVM classifier, there was a small improvement from 0.598 to 0.61.

After testing multiple iterations and removing word length restrictions, the same pattern emerged for other training data.