Twitter Sentiment Classifier Report

I. Background

Nowadays, social media is playing a significant role in most people's daily life. Billions of opinions and ideas are shared on popular social media platforms such as Twitter and Facebook every day. People express happiness, sadness, madness and all other kinds of emotions on these platforms. These emotions could become very useful information for researches such as how people think or react to a product, a celebrity or a piece of news. The ideal situation is that we could apply natural language processing skills to a dataset of, for instance, a pile of Twitter messages and justify the corresponding sentiment as the output. The research in this direction, however, still faces many unsolved problems at the current stage.

This is why the SemEval-2017 competition Task 4 is carried on, of which this exercise is under the context. The competition is delivered as a part of the International Workshop on Semantic Evaluation and has been running annually since 2013. The whole task focuses on sentiment analysis in twitter data and we are particularly interested in the first subtask, which is message polarity classification.

In this exercise, we would like to classify whether a given Twitter message is of positive, negative, or neutral sentiment, based on a training dataset with already justified sentiments. We are given one training dataset, one development dataset in a smaller size and three test datasets for testing the classifiers. They include Twitter messages on a wide range of topics, including a mixture of entities, products and events from different time periods. The task is separated into the following steps: preprocess all the data to remove unwanted characters and also to avoid missing any important information; extract semantic features from the training dataset and the test datasets; build 3 different classifiers and evaluate them using the development dataset; apply the final classifiers to the test sets; discuss and evaluate the performance of the model in terms of accuracy and macro-average balanced F1 scores.

II. Data Preprocessing

Before extracting features from the training data, firstly we need to preprocess the data. Some codes are reused from the first exercise. The following preprocessing steps have been done in the given order:

- All the characters are encoded in the utf-8 coding system. This requires manual changes due to system restrictions of the Apple MacBook used and the same process is done to both the classification.py and the evaluation.py;
- All the characters have been lowercased, noting that this only affects the alphabetic characters;
- The original data files are formatted as TSV (tab-separated-values), with one tweet per row including values in the format of "tweet-id<tab>sentiment<tab>tweet-text". What we would like to analyse is the text part and this format is hard for evaluation. I therefore separate them by tabs and save each line in a list with three elements, tweet ID, sentiment and tweet text respectively;
- There are some special characters in the original training set which are still in the Unicode format after encoding. For example, '\u2019', which should be a single quotation mark, was still written in such a Unicode format and are then replaced by the actual punctuations;

- There were many tweets with URLs in the content in order to redirect to certain webpages. They are not helpful in terms of sentiment analysis and thus are removed like how it has been done in exercise 1;
- People always include “RT” to retweet a tweet. This is also irrelevant to the sentiment and is removed;
- Another particularity of tweets data is that there will be hashtags like #something and user mentions like @someone. This feature is also removed from the data;
- People may use some elongated words in tweets to emphasize emotions, such as “coooooollllll” and “looooooove”. They are very important for sentiment analysis and should not be removed. However, different people may type different number of “o”s for “love”. I therefore removed extra letters and change the elongated words to the simplest format;
- After the above preprocessing steps, all the non-alphanumerical characters are also removed, since they do not influence the sentiment (some characters like exclamation marks can express either positive or negative sentiments and thus become not important);
- All-digit words are also removed used the method in exercise 1;
- I have also removed stop words in English, but keep some of them which may influence the sentiment, such as 'over', 'under', 'below', 'more', 'most', 'no', 'not', 'only', 'such', 'few', 'so', 'too', 'very', 'just', 'any' and 'once'.

After the steps above, the preprocessing part of this task has been finished. There are some factors not added in the current analysis. Firstly, when people post a Twitter text, they are likely to include some Twitter-special abbreviations or lingos due to the character limits of Twitter, such as “ICYMT” which means “In case you missed it.”, “DM” which means “direct message”, etc. This is not done in this task because I have not found an existing file which include all the lingos of Twitter. Secondly, emojis and emoticons are removed in this task because of the encoding issues and the lack of dictionaries of emoticons. This could be considered for future analysis.

III. Feature Extraction and Classifier Building

The semantic feature I choose to use in this task is ngrams. Initially, I only wanted to use unigrams (i.e. single words), but the results were not satisfying. Unigrams could not well represent the characteristics of the training dataset. The accuracy and the F1 scores were quite low. This is partially because double words or triple words may completely change the sentiment of the text, such as “not good” and “not at all”. Therefore, I decided to use bigrams and trigrams as my feature along with unigrams.

There are still some problems with the feature. Figure 1 and Figure 2 show the word clouds for positive and negative bigrams in the training set based on frequencies. They are drawn in python with the help of a package called wordcloud. It can be noticed that some bigrams like gonna (going to), may not, etc. have high frequency disregards to the sentiment of the message. It can also be noticed that there are many entity-related words. For example, National hot dog day, national ice cream day, the movies “ant man” and “foo fighters” appear many times in positive tweets and names like Donald Trump and Boko Haram are more likely to appear in the negative ones. This may decrease the usability of our features, as different texts are in different time periods and the corresponding hot topics may be different entities. This can also be evaluated in future analysis.

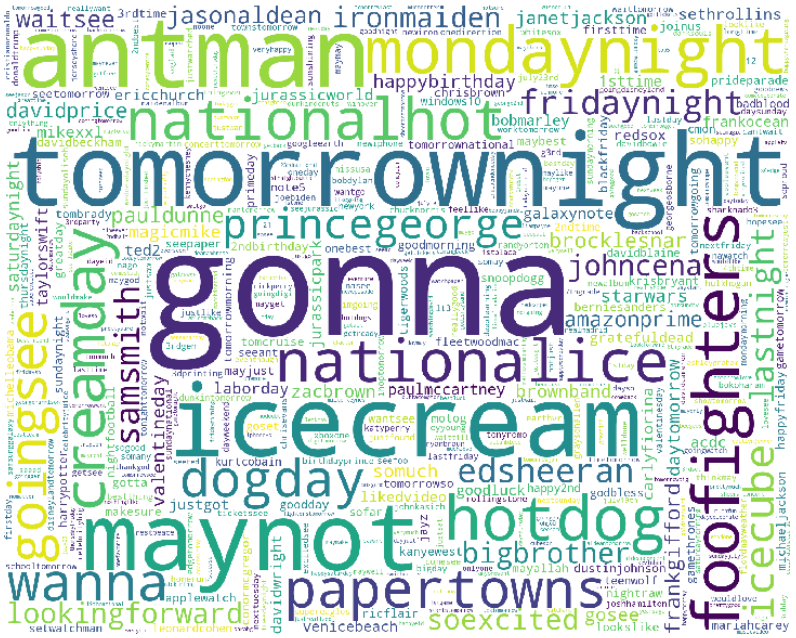

Figure 1. Wordcloud for Positive Bigrams in the training set

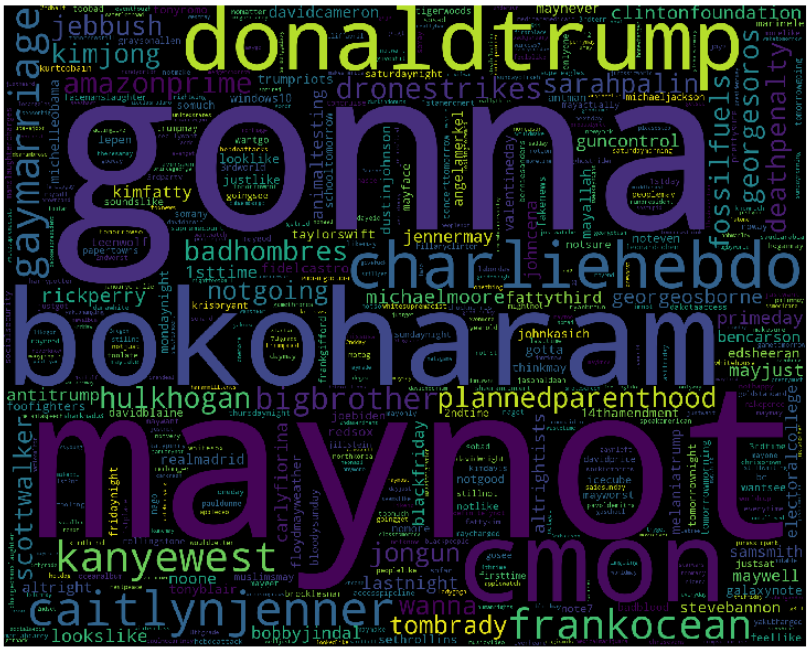

Figure 2. Wordcloud for Negative Bigrams in the training set

We would then like to develop 3 classifiers and put the obtained feature in them. For the classifiers, we import classifier construction functions from the nltk.classify package in python. In this task, three different classifiers are selected, Naïve Bayes classifiers, Maximum Entropy (MaxEnt) classifier and logistic regression classifier. The Naïve Bayes classifier is based on the classic Naïve Bayes algorithm. The Maximum Entropy classifier is based on the maximum entropy modelling. It considers all the probability distributions which are empirically consistent with the training dataset and then choose the one with the highest entropy. The algorithm used for the training data is Generalized Iterative Scaling (GIS) algorithm. Other

algorithms like Improved Iterative Scaling (IIS) algorithm were tried as well, but did not have as good performance as the GIS one. The logistic regression classifier is based on the logistic regression model. Results and evaluations are shown in the next section.

IV. Evaluation and Conclusion

Firstly, we apply three classifiers to the development set to evaluate them and to avoid overfitting. The results are shown in Figure 3. It is shown that the logistic regression model has the best performance on the development data and thus is applied to the test datasets, whose results are shown in Figure 4.

```
Training NaiveBayesClassifier
accuracy: 0.3655
twitter-dev-data.txt (NaiveBayesClassifier): 0.619
             positive  negative  neutral
positive     0.610     0.061     0.329
negative     0.109     0.530     0.361
neutral      0.235     0.100     0.665

Training MaxentClassifier
accuracy: 0.1865
twitter-dev-data.txt (MaxentClassifier): 0.272
             positive  negative  neutral
positive     0.732     0.048     0.220
negative     0.125     0.875     0.000
neutral      0.265     0.218     0.517

Training LogisticRegressionClassifier
accuracy: 0.373
twitter-dev-data.txt (LogisticRegressionClassifier): 0.628
             positive  negative  neutral
positive     0.611     0.056     0.333
negative     0.100     0.554     0.346
neutral      0.224     0.111     0.665
```

Figure 3. Results for development dataset with 3 classifiers

```
Training LogisticRegressionClassifier
accuracy: 0.41772868875672614
twitter-test1.txt (LogisticRegressionClassifier): 0.546
             positive  negative  neutral
positive     0.603     0.073     0.324
negative     0.172     0.573     0.254
neutral      0.260     0.156     0.584

accuracy: 0.328656233135456
twitter-test2.txt (LogisticRegressionClassifier): 0.574
             positive  negative  neutral
positive     0.678     0.063     0.259
negative     0.196     0.491     0.313
neutral      0.356     0.089     0.555

accuracy: 0.43421605716687683
twitter-test3.txt (LogisticRegressionClassifier): 0.535
             positive  negative  neutral
positive     0.622     0.080     0.299
negative     0.216     0.483     0.302
neutral      0.314     0.125     0.561
```

Figure 4. Results for test datasets with final classifier

As the above figure show, accuracy, macro-averaged F1 score and confusion matrix are calculated for evaluation. Accuracy is (number of correct classifications)/ (the number of total tweets). The macro-average balanced F1 score is $2\times\frac{precision\times recall}{precision+recall}$. The diagonal entries of the confusion matrix show the proportion of items which are classified correctly. Other entries show the proportion of wrongly classified items with corresponding type of errors.

From Figure 3, we can tell that the MaxEnt classifier performs badly on the development set, with very low accuracy and very low F1 scores. Naïve Bayes classifier is better than the MaxEnt, but still has lower accuracy and lower F1 score compared with the logistic regression classifier. Therefore, we selected logistic regression classifier along with the combination of unigrams, bigrams and trigrams features as the final classification model. The model is then applied to three test sets.

The results have considerably good F1 scores, but the accuracy is in general not very high. As discussed in the previous sections, more rigorous preprocessing process and attempts on other features such as word embedding should be considered in future studies and may then increase the accuracy and the F1 scores.