Twitter Sentiment Analysis Milestone Report 2

Text Cleaning

A tweet needs to be cleaned to help improve the learning efficiency of machine learning models. Tweets contain punctuation, stopwords, and combination of lower and uppercase words, which affect the model's learning capability. The following are the steps used in cleaning the tweets:

1. Convert text to lower-case: Converting tweets to lowercace because text analysis is case sensitive. This means that "Good" and "good" are considered two different words by the model.
2. Remove URLs: Remove all possible URL links since we are not going to dive deeper into the links
3. Remove usernames: Remove all possible usernames since we are not going to dive deeper into the usernames
4. Remove punctuation: Punctuation is removed from data because it does not contribute to text analysis. It impairs the model's ability to differentiate between punctuation and other characters.
5. Remove stopwords: Stopwords have no analytic value in text analysis. They need to be removed to reduce complexity of the model.

After cleaning the text we can see the the words that are most common in each sentiment in the following word balloons:

Figure 3 Negative Word Balloon

Figure 4 Positive Word Balloon

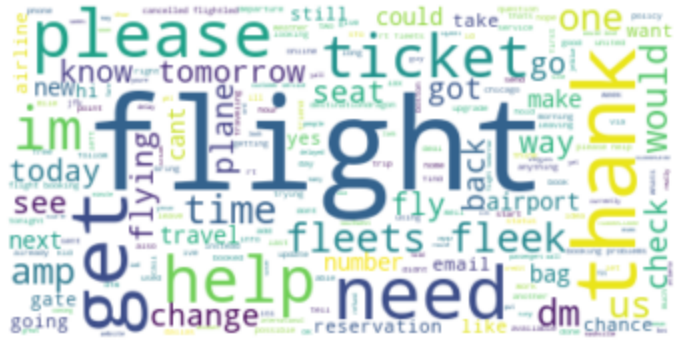

Figure 5 Neutral Word Balloon

The word balloons above show which words show up the most in each type of sentiment. The word ‘flight’ is an often used word in all three sentiments. Negative sentiments contain: customer service, help, plane, and bag. On the positive sentiment word balloon we see ‘thank’ as the most prominent word. Neutral sentiments have please, need, and get as prominent words.

These word balloon graphs show a snapshot of prominent words that will likely be used in the models.

Model

Now that the data is cleaned and organized, a predictive model may be built with the sklearn algorithms. The model will be trying to predict the *airline_sentiment* using text from the tweets. Classifier machine learning algorithms will be used.

The text data will be put into vectors to build a vocabulary for analysis. The two vectorizers are CountVectorizer and TfidfVectorizer. CountVectorizer provides a simple way to both tokenize a text and build a vocabulary of known words. It also encodes the text using the vocabulary it built. TfidfVectorizer calculates word frequencies. It will tokenize, learn the vocabulary, inverse text frequency weightings and encode new text.

After the vocabulary vectorizers are built the data will be scaled using MaxAbsScaler. MaxAbsScaler scales and translates each feature individually such that the maximal absolute value of each feature is the training set will be 1.0. It does not shift/center the data, and thus does not destroy any sparisity. Scaling helps to not let features with larger units have undue influence on the classifier as would be the case if the classifier uses some sort of distance measurement as a similarity metric.

After scaling the data, classification algorithms will be used to predict the twitter sentiment. Three algorithms will be tested. Naive Bayes, SVC, and Random Forest. Each of these algorithms will also be tuned with cross validation to find the best algorithm parameters for the model.

The scoring parameter the models will use is F1. F1 score conveys a balance between precision (number of positive predictions divided by the total number of positive class values predicted) and recall (number of positive predictions divided by the number of positive class values in the test data). A higher F1 score means the model is better at prediction. Each model during the

Randomized Search and Grid Search Cross Validation will give a F1 score and a mean and standard deviation will be taken of each F1 score of all models.

The text data was split into training and testing sets and kfold was used to split the training set into training and validation sets. After running all the different models with Randomized Search and Grid Search cross validation results were obtained. The models with the best mean F1 score and mean standard deviation were used to predict the test set. The following table shows the F1 scores for the training and validation sets of each model followed by the metrics accuracy, precision, recall, and F1 score of the test sets:

	train-score	validation-score	accuracy	precision	recall	f1
CountVectorizerMultinomialNB	0.969 (0.002)	0.634 (0.009)	0.739	0.688	0.617	0.641
CountVectorizerSVC	0.968 (0.001)	0.707 (0.01)	0.774	0.731	0.699	0.713
CountVectorizerRandomForestClassifer	0.991 (0.001)	0.701 (0.012)	0.755	0.690	0.688	0.689
TfidfVectorizerMultinomialNB	0.985 (0.001)	0.639 (0.009)	0.735	0.669	0.624	0.640
TfidfVectorizerSVC	0.88 (0.001)	0.695 (0.005)	0.749	0.696	0.713	0.703
TfidfVectorizerRandomForestClassifer	0.99 (0.001)	0.698 (0.011)	0.761	0.707	0.694	0.700

Table 1

From the table above we see that using the CountVectorizer and SVC algorithm had the highest scores in the majority of categories. It had an accuracy of 77.4% and an F1 score of 0.713. These are good scores even with the data being unbalanced in favor of negative sentiment.