Predicting the Origin of a Tweet

Davis Busteed — ECON 484

## 

Table of Contents

[INTRODUCTION 1](#_Toc26881598)

[DATA 1](#_Toc26881599)

[Data Source 1](#_Toc26881600)

[Data Description 2](#_Toc26881601)

[METHODS 4](#_Toc26881602)

[Approach #1 4](#_Toc26881603)

[Approach #2 4](#_Toc26881604)

[Approach #3 5](#_Toc26881605)

[RESULTS 6](#_Toc26881606)

[Approach #1 6](#_Toc26881607)

[Approach #2 7](#_Toc26881608)

[Approach #3 7](#_Toc26881609)

[CONCLUSION 8](#_Toc26881610)

[Appendices 10](#_Toc26881611)

[Appendix A – Data Collection Workflow 10](#_Toc26881612)

[Appendix B – Data Summary 11](#_Toc26881613)

[Appendix C – Additional Box Plots 12](#_Toc26881614)

[Appendix D – Models Evaluated with Extracted Features 13](#_Toc26881615)

## INTRODUCTION

Wide-spread social media usage has made unfathomable amounts of unstructured data available to the world. For this project, I wanted to explore the feasibility of “adding structure” to the millions of tweets that are posted daily. Specifically, I became interested in classifying the nationality of Twitter users based on the text of their tweet. For many international Twitter users, country of origin could easily be determined by looking at the language of the tweet. Because this strategy is infeasible for countries that speak the same language, I decided to begin my research by developing a model that would classify whether tweets written in the English language originated from the United States or from England.

A classification model that accurately predicts whether a Twitter user is American or English could be very useful to many organizations. For example, firms whose markets span both countries may be interested in classifying their prospective customers for the sake of marketing. Depending on the accuracy of the predictive models, this project could potentially be used to create new datasets that could be included in future research projects. Not all Twitter users have location services activated, so accurate data concerning the origin of a tweet isn’t easily attainable through normal data collection processes.

## DATA

### Data Source

I used Tweepy (Twitter’s API for Python) to collect English and American tweets. Using the latitude and longitude coordinates provided by Twitter users who have enabled location services, **get\_tweets.py** wrote tweets to text files that corresponded to the United States and England. After this, **txt\_to\_json.py** combined the separate text files into a single JSON file. Lastly, **feature\_extraction.py** used this JSON file to create numeric features from the text of the tweets, and saved the final data file to processed\_tweets.json. See Appendix A for a detailed workflow of the data collection process.

### Data Description

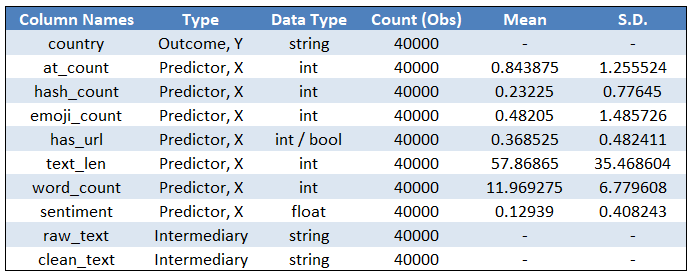
The dataset contains 40,000 observations (20,000 from each country), and was collected in small portions over several days to obtain a more representative sample. Table 1 lists the variables that make up the dataset, including the outcome variable (*country*), and the model predictors that were derived during the feature extraction process of data collection. See Appendix B for a detailed summary of each variable.

Table 1 – Summary statistics for model outcome and predictors

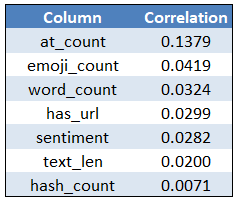
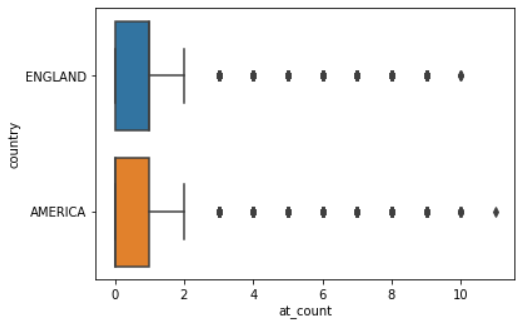
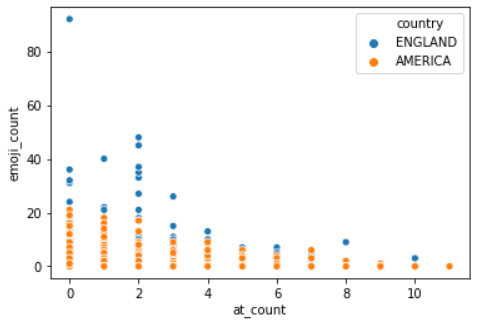
Before beginning the machine learning process, I attempted to find relationships among the dataset by looking at summary statistics and correlations (see data\_exploration.ipynb). Among the model predictors, *at\_count* had the highest correlation with the country label, although it didn’t have a large magnitude (see Table 2). Unfortunately, a box plot of *at\_count* for each country didn’t provide me with any unique insights; it seemed that the average *at\_count* was similar for both regions, and that each had its share of outliers (see Figure 1). Appendix C contains similar box plots using other variables, in which the distribution of each predictor does not vary much between the two countries. One insight I found as I iterated through different visualizations was that some tweets from England have a higher number of emoji. The box plot of *emoji* and *country* showed this trend as well, but the scatterplot in Figure 2 emphasizes the difference between the two countries.

Table 2 (left) – Correlations between model predictors and response variable (country)

Figure 1 (right) – Boxplot for the number of user mentions (at\_count) for each country

Figure 2 – Scatterplot of at\_count and emoji\_count for both countries

## METHODS

Taking a closer look into the data did not provide me with any hints on which classification algorithms would give the best predictions, so I decided to experiment with several classification approaches and see which ones had the best out-of-sample fit.

### Approach #1

To streamline the creation and evaluation of multiple predictive models, I developed a simple framework which would take a series of predictive algorithms along with its possible tuning parameter values, and select the model with the best out-of-sample fit (see ModelEvaluator in model\_evaluation.ipynb). For each classification algorithm, cross validation and tuning parameter selection were performed by Scikit-learn’s GridSearchCV function. After which, these parameters were used to create a model in which out-of-sample fit was calculated.

Using this framework, I began creating and evaluating a series of algorithms, using all of the numeric features extracted from the tweet. Trees, regressions, and neural networks were among the different algorithms tested. See Appendix D for screenshot of the model specifications. Out of the models tested, the Gradient Boosting algorithm proved to have the highest test accuracy. Detailed results of this stage of evaluation can be found in the next section.

### Approach #2

After this, I experimented with other approaches outside of standard feature extraction. The first of which was using term the frequency-inverse document frequency (TF-IDF) algorithm to create a matrix from the text found within the tweets. This matrix gave “importance scores” for every word found in the sample, and could easily be used as predictors in the classification algorithms I previously used. Figure 3 shows a small subsection of the TF-IDF matrix.

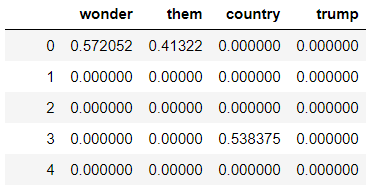
Because each unique word in the entire sample became a column in the TF-IDF matrix, I used PCA to reduce the predictor range from ~30,000 to 100. Using the Gradient Boosting model selected in the previous step, I ran two separate models with the TF-IDF matrix. One used the first 100 principal components of the TF-IDF matrix as model predictors, and the other used the previously used extracted features along with the 100 principal components. Results for these classification models can be found in the next section.

Figure 3 – Subsection of TF-IDF matrix

### Approach #3

For my final approach, I took a step away from the Scikit-learn framework and used MALLET, a linguistical machine learning tool developed by UMass Amherst.[[1]](#footnote-1) MALLET’s key strength is document classification, so feature extraction and natural language processing is encapsulated within the toolkit. I experimented with several algorithms with MALLET, including the Decision Tree, Winnow Classifier, Naïve Bayes Classifier, and Maximum Entropy Classifier. Tuning parameters for the different models were chosen with cross validation, and out-of-sample testing was conducted by splitting the data into a training and test set. The following section provides the results from models created with this framework.

## RESULTS

### Approach #1

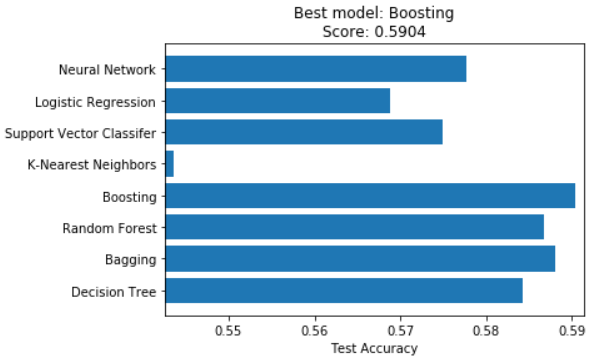
As mentioned in the previous section, the ModelEvaluator framework provided functionality for evaluating the out-of-sample fit for each model. Because each model is only built with the training set of the data, the test set could be used to estimate how the model would perform with “unseen” data. Figure 4 shows the out-of-sample fit (test accuracy) for each classification algorithm that was evaluated in the process explained previously.

Figure 4 – Comparison of different algorithms using extracted features

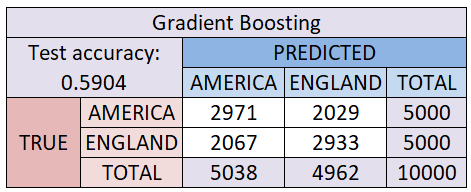
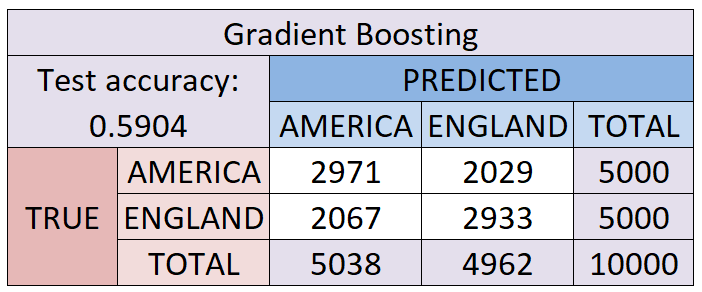
With a test accuracy of 0.5904, the Gradient Boosting model had the best out-of-sample fit. Despite outperforming the other models, a test accuracy of 59% suggests that this model would only do “9% better” than a random guess at whether the tweet originated from the United States or England. As seen in the confusion matrix (Figure 5), the model gives better predictions than a perfect 50-50 split, but it still lacks the accuracy required for credibly adding country labels to tweets outside of the dataset.

Figure 1 – Confusion matrix for Gradient Boosting model

Figure 5 – Comparison among models using TF-IDF principal components

### Approach #2

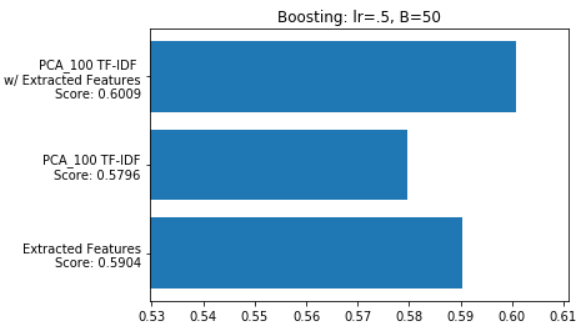
To measure the predictive power of the first 100 principal components of the TF-IDF matrix, I compared the test accuracy of the Gradient Boosting model using just extracted features, the model using the TF-IDF principal components, and the model using both. As shown in Figure 6, the Gradient Boosting model that combines TF-IDF and the original extracted features gave slightly better predictions than the model without TF-IDF. Despite this, it is worth considering whether or not this 1% increase in accuracy is worth the training time increase of ~20 minutes.

Figure 6 – Comparison among models using TF-IDF principal components

### Approach #3

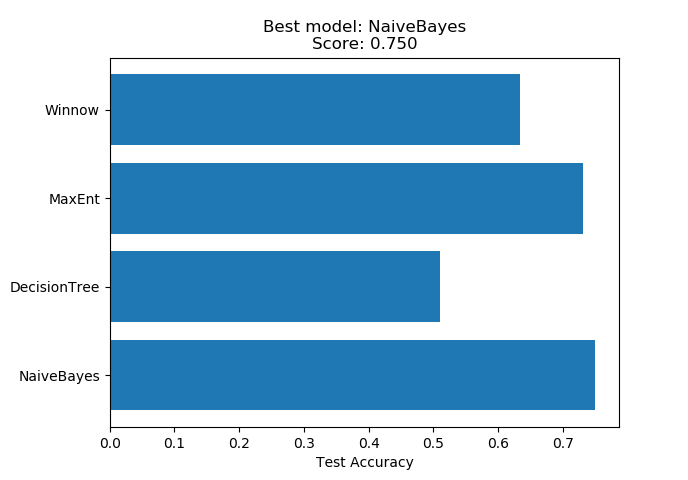
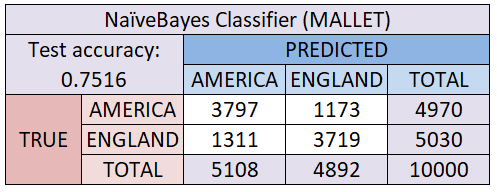
As I stepped away from Scikit-learn for my final approach, I was met with better results. In a similar process from before, I was able to gather all of the out-of-sample fits for each algorithm tested and compare the results (see Figure 7). The Naïve Bayes Classifier had a test accuracy score of 0.7516, which is quite an improvement from the Gradient Boosting model. By looking at the confusion matrix shown in Figure 8, it seems that the model performs similarly for predicting both classes. With a test accuracy of 75%, there is a significant advantage of using the model compared to relying on random guesses.

Figure 7 (left) – Comparison among models created with MALLET

Figure 8 (right) – Confusion matrix for Naïve Bayes Classifier

## CONCLUSION

If limited to the efforts of this project, I would select the Naïve Bayes Classifier developed with MALLET framework, because it had a test accuracy score of 75%, which was much higher than the Scikit-learn approaches using extracted features and TF-IDF. Despite this, I still think that this project would need more work before its results could reasonably be used in “adding structure” to tweets by predicting their country of origin.

I think the primary limitation of all three approaches was the lack of sizable features found in the tweets. By design, tweets are extremely small. Although the sample size was large, each tweet could only contain a limited amount of data, making feature extraction difficult for both the Scikit-learn and MALLET approaches.

Moving forward, it would be interesting to filter out shorter tweets in the machine learning process, and develop a predictive model that focuses on tweets with more content. If this approach improved the model accuracy, it would also be interesting to expand the research to including other English-speaking countries such as Canada.

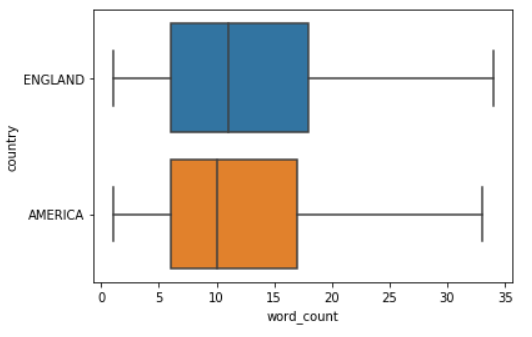
## Appendices

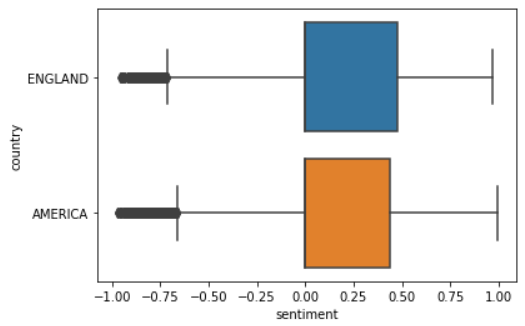
### Appendix A – Data Collection Workflow

### Appendix B – Data Summary

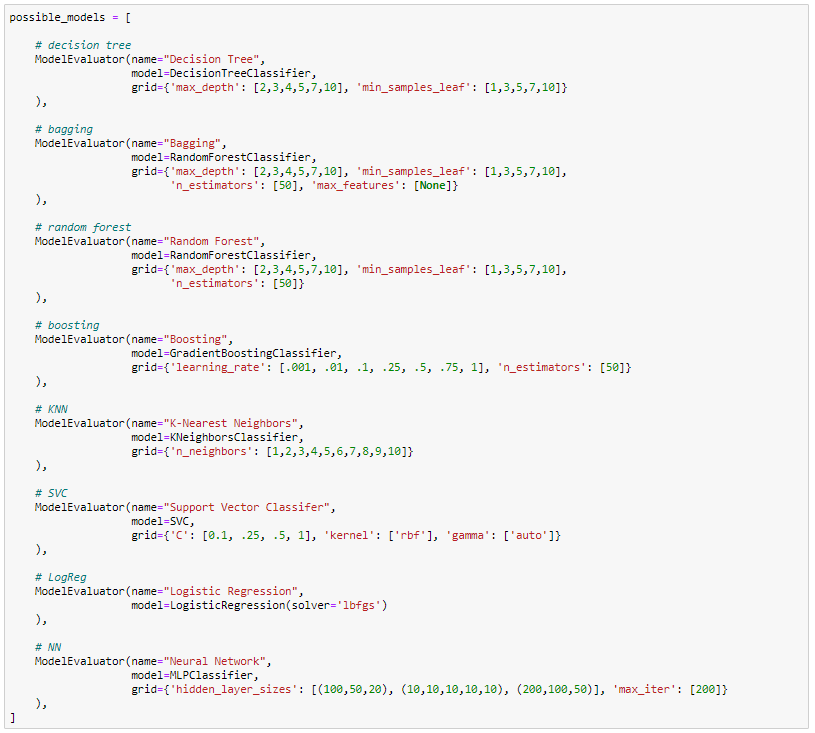
|  |  |  |
| --- | --- | --- |
| Name | Description | Notes |
| country | The country of origin of the tweet, either ‘AMERICA’ or ‘ENGLAND’ | This is the y-variable, the value to be predicted |
| raw\_text | The raw, unaltered text of the tweet, as recorded from the Twitter API |  |
| at\_count | The number of user mentions, in which a tweet specifies another Twitter account with the @ symbol | As per Twitter standards, a user mention is defined as the @ symbol followed by text. A stand-alone @ symbol doesn’t contribute to the “at\_count” |
| hash\_count | The number of hashtags (#), Twitter’s topic identification feature | Same as above, except with the # symbol |
| emoji\_count | The number of emoji used within the tweet |  |
| has\_url | Whether or not the tweet had an embedded URL, in which a tweet can be linked to an external source | 0 = no URL in tweet  1 = URL in tweet |
| clean\_text | The “pure” text of the tweet. Derived by removing @s, #s, URLs, emoji, and irregular spacing from “raw\_text” | Irregular spaces include multiple spaces and tabs, both of which were substituted with a single space |
| text\_len | The number of characters in the “clean\_text” |  |
| word\_count | The number of words in the “clean\_text” |  |
| sentiment | The sentiment score of the “clean\_text” | NLTK’s VADER, which was trained on social media text, was used to calculate scores. Ranges from -1 to 1 |

### Appendix C – Additional Box Plots





### Appendix D – Models Evaluated with Extracted Features



1. http://mallet.cs.umass.edu/ [↑](#footnote-ref-1)