Kaggle Team: Harrison Unruh

Aaron Bu (acb256), Harrison Unruh (hju3), Cindy Wang (cw653), Zihan Zhang (zz229)

Score: 0.81666

Preprocessing Techniques

We used a few preprocessing techniques for our data. For our sentiment analyzer, we stripped each tweet of punctuation, and converted all the text to lowercase. We also converted the time each tweet was posted to an integer that represented the number of seconds past midnight each tweet was posted.

What We Learned While Exploring the Data

Our intuition from prior knowledge of Trump and his tweeting was that the most useful characteristics of the data would be the frequency of certain words (like “Hillary”, “fake news”, etc.). We also expected tweets to be more likely from Trump versus his staffers if they were posted at times outside the normal workday or with strong sentiment. Ultimately we found that the most useful feature was whether or not links were included in tweets: links were overwhelmingly present in tweets from iPhone, but not from Android. Replies to other tweets were helpful as well, and generally indicated that the tweet was sent from Android. Sentiment provided some added value, but ultimately wasn’t a particularly strong indicator. Posting time was one of the more heavily used features in our model, but was not nearly as useful as links or replying to other tweets.

How We Selected Our Model

We selected the random forest classifier mostly because it has a reputation for being reliable and relatively straightforward. Additionally, we tried out many other classifiers, including naive

bayes, adaboost, SVM, and ERM, before settling on this one because it predicted the best results. In moving forward with the model, we tested out a myriad of features before finding a combination that allowed for the most accurate predictions.

Features Extracted

We extracted quite a few features from the dataset. The bulk of our features were derived from the counts of various character sequences. We used the sklearn CountVectorizer to count the frequencies of character n-grams, limited to word boundaries. We then applied a TfIdf transformer to normalize n-grams by their distribution across the dataset. An NLTK pre-computed sentiment analyzer was used to evaluate the positive, negative, and neutral sentiment intensity of each tweet (rated between 0 and 1). We also directly extracted the time of the tweet, number of retweets, and number of favorites, provided by the dataset.

Handling Hyper-parameters

We used a random forest classifier, largely with the default parameters. The only parameter we adjusted was the number of trees in the forest, which we chose by trying different forest sizers. We found that prediction accuracy peaked around 20 trees, and dropped with greater or fewer trees.

Code Cited

We used the website below to get our code started, to learn more about sk-learn, for general strategy, and for sk-learn syntax.

https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

We also used this website to learn how to conduct sentiment analysis.

http://www.nltk.org/howto/sentiment.html