

Kaggle Competition Report

Xingyu Chen, Yang Liu
xc374, yl572

This project is to identify the device the Trump used to write tweet with. The problem could also be interpreted as if Trump is truly the author of tweet on his account or the tweets are posted another person on behalf of him. Thus, the semantic information conveyed in the tweet are vital source to explore the features and external information like favourite count and retweet count might also be important.

Features

Other than the favourite count and retweet count in the original dataset, Here are some features we extracted:

The first three features we extracted are whether the tweet contains hashtag(#), at(@), or external link, because it is possible that Trump likes plain text without those special characters. Due to the same reason, we also checked if the tweet contains punctuations such as exclamation mark, quotes, etc. We also believe the punctuations in each tweet might reveal some information like quotation mark and exclamation marks, etc. Therefore we count the occurrence of different punctuations and mark them as features as well.

Also, to capture the vocabulary used by Trump perform TF-IDF on the tweet text. We also record length of feature. Then, we filtered the sentence by deleting all the noisy marks like punctuations (already counted them) and then extract the positive and negative sentiment from the tweet using NLTK sentiment analysis.

Besides language patterns, we also categorize the timestamp of tweets into different categories: whether the tweet is sent on weekdays, the different time slots the tweet is sent (multinomial feature), since we believe that different people like to use their phones in different time.

Finally, we did something similar in project 3: we split sentences into tokens and hashed tokens into fifteen buckets. How we choice bucket size is explained in later section.

Preprocessing

In our training data, some features represents counts, some represents booleans, and some represents timeslots. If we feed this training data into model, some features will be over-weighted heavily. We assume that each feature shares the same weight, so we normalize each feature before we feed the data. We divided the data in the training file into our training set and validation set with a ratio of 0.85 : 0.15.

What we learned while exploring the data:

By examining the data, we found that some features are important, like text and time, while some features are not useful in prediction, such as favorited and truncated. Also, punctuation might not look redundant and they might reveal some patterns of using different devices.

How we searched for hyper-parameters:

We basically iterate through possible hyperparameters and choose the one with the best performance. Take the max_depth of random forest as an example. We printed out the validation accuracy of the forest as a function of max depth, and found that max depth 5 resulted in the highest accuracy.

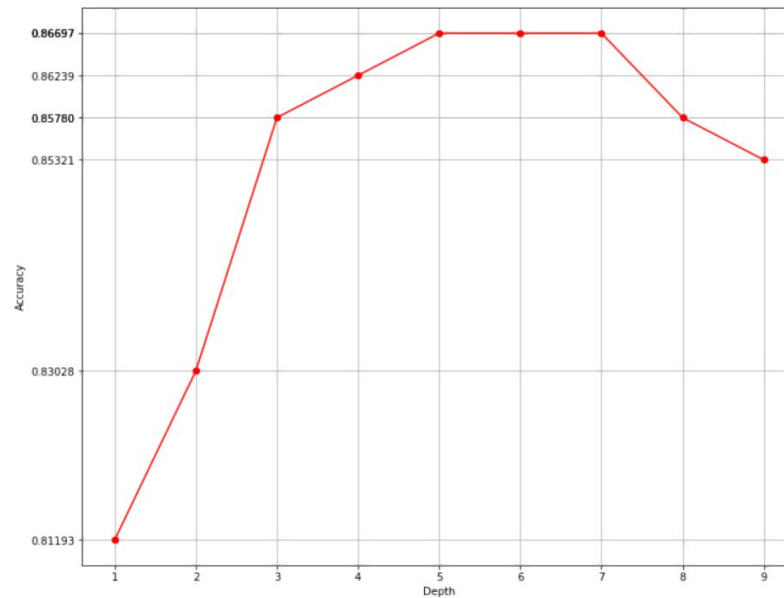


Figure 1. The validation accuracy with respect to tree depth for random forest classifier

How we selected our model:

We have tried different prediction models and choose one with the highest validation error.

Machine Learning models:

The tuning process of parameters is done by grid search as explained below.

The best performances of all different classifiers are noted below after tuning. It should be noted for SVM and NB to perform well on this dataset, we used PCA to extract 3 most important principle components.

KNN and Random forest and AdaBoost classifiers achieve the same accuracy for the validation set.

Model	KNN (k=3)	Random Forest (depth=5)	AdaBoosting	Gradient Boosting	SVM (linear kernel)	PCA+SVM	NB	PCA+NB
Accuracy	0.867	0.867	0.867	0.858	0.587	0.807	0.5504	0.802

Table1. Summary of Machine Learning Models

Deeping Learning Model: Neural Network

Essentially, we need to model dependency of created features to the target label. Due to the size of feature we created, feature reduction is needed to remove unnecessary features. We have tried selecting features using Ridge Regression via shrinking feature coefficients. For the same purpose, Neural Network will extract the key feature and assign a reasonable weight to it automatically and thereby might generating a better representation of model to optimize the result for us.

Layer (type)	Output Shape	Param #
dense_23 (Dense)	(None, 256)	7936
dropout_12 (Dropout)	(None, 256)	0
dense_24 (Dense)	(None, 2)	514

Total params: 8,450
Trainable params: 8,450
Non-trainable params: 0

Train on 1034 samples, validate on 55 samples

Epoch 1/3
1034/1034 [=====] - 2s 2ms/step - loss: 0.4246 - acc: 0.7979 - val_loss: 0.3779 - val_acc: 0.7636

Epoch 2/3
1034/1034 [=====] - 1s 1ms/step - loss: 0.3158 - acc: 0.8675 - val_loss: 0.3154 - val_acc: 0.8182


Epoch 3/3
1034/1034 [=====] - 1s 1ms/step - loss: 0.3094 - acc: 0.8607 - val_loss: 0.2459 - val_acc: 0.9091


Figure 2. Model architecture and Training process

To have more data, for the final model, we combine the training and validation set. One downside of that, due to lack of dev set, during training, we have no idea if we overfitting the training set. Therefore, we only trained 3 epoches, given the neural model is very shallow to avoid overfitting. Also, we add dropout after the first layer for the same purpose. The validation accuracy of our model achieves **0.871** for the neural model we created.

Kaggle Score:

The accuracy of text set is 0.858.

12 — **tooMuchMogic**  0.85833 25 1d

Your Best Entry 

Your submission scored 0.84166, which is not an improvement of your best score. Keep trying!