(1)
Here's an overview of our data pipeline.

We extract data from 3 different sources: tweets of Donald Trump and Hillary Clinton from Tweeter APIs, Presidential election-related Data from Kaggle, Campaign and other speeches from Internet.
Then, we convert them into uniform format using our adapter.
To make it more realistic for general cases, as well as to increase the difficulty and make our classifier more robust, we then dirty our corpus by introducing typos at a random rate.

We then use Hidden Markov Model to correct those typos. Each state will be a correct character, and each emission is

an observed character. We build a transition model based on numerical rate and we build an emission model based on keyboard structure. For instance, if two characters are adjacent on the keyboard, then it will have a higher probability than a random pair.

In next step, we will input our processed data into our classifier.

Finally, we would love to also generate similar style texts using our trained RNN.

(2) Here's a summary of all the approaches we use in our classifier, which we will talk in more details later.

(3)Naive Bayes: Bag of Words

Let's first recall a what a general Naive Bayes Model is.
Given a data point, with a bunch of features F, we want to predict its class Y.

For bag-of-words representation, a feature is a word. We estimate the conditional probability table of P(Y) and P(FIY) in the model by counting the occurrences of each word in each class.

The graph on the right contains some interesting results we found, which could potentially become our features.

(4) For Feature-based Approach:

Instead of using a word as a feature, we consider customized features like average word length, sentence length and so on.

Each feature is characterized as a real value.
For each feature and each class, we use all values in all data points to generate a normal distribution to estimate the probability table for P(FIY) in the model.
Then, for each given data point with a set of features, we estimate the probability using the corresponding normal distribution.

(5) HMM classification
HMM is actually a pretty popular method for sentiment classification, which could be a similar problem.
Right now, we are still in the process of exploring this HMM classification method. There are some interesting problems we will have to solve. How are we going to

model transition and emission models? Besides empirical rate or maximum likelihood, domain specific knowledge could also play an relatively important role here.
Furthermore, we may also try to explore a HMM and SVM hybrid method discussed in papers we are currently reading.

(6)Possible Improvements.
There are still a lot of possible improvements we could try in our next step.

First, we would love to think of ways to retrieve more data. We could possibly in the future also reuse our generated text as our input.

Second, we would like to explore feature selection using simulated annealing, since a lot of times, we don't want to import too many features in our classifier, not only for speed issues, but also for potential overfitting issues. Currently, our bag-of-words is 93% accuracy, whereas feature-based approach is not as good.

Third, we would love to explore variations related to RNN. For instance, instead of using word2vec representation, we could try Global vectors of word representation.

Lastly, we would love to explore nearest neighbor classification.

and convert them into uniform format using our adapters. Currently, we then directly apply our classifier. In the future, we would love to dirty our current corpus (like

First, we attempted the simplest method using Naive Bayes classification with bag-of-words representation. In this case, a feature F in Naive Bayes is a word W. We can first calculate all the conditional probabilities in the form P(W l class). Recall the approach of naive bayes. Then, given arbitrary sequence of words, we can apply the following equation, and normalize it to get the result.

Secondly, we used a feature-based naive bayes approach.

For each class, consider all the values for each feature. We can use their mean and variance to get an estimated normal distribution. Then, for an arbitrary sequence of words given some feature value, we can use the corresponding normal distribution to estimate the conditional probability P(word l class).

Notice that here, features values can be real numbers.

We first want to estimate the conditional probability tables for P(Y) and P(FIY) using empirical rate.
Then, for a testing data point, we calculate the joint distribution using our parameters. And finally, normalize it to get a desired probability for each class.
We first want to estimate the conditional probability tables for P(Y) and P(FIY) using empirical rate.
Then, for a testing data point, we calculate the joint distribution using our parameters. And finally, normalize it to get a desired probability for each class.