'Speech' Recognition: Using NLP to Attribute Quotes to Speakers

LING 227 Final Project

Luke Benz
Will Langhorne
Kevin Truong

December 15, 2017

Introduction

History remembers great speakers for their idiosyncracies. From JFK's authenticity and passion to Donald Trump's divisive rhetoric, U.S. presidents are particularly identifiable by their speeches. While nearly any citizen can name the presidents who said "Nothing to fear but fear itself" (Franklin Delano Roosevelt), "Ask not what your country can do for you–ask what you can do for your country" (John F. Kennedy), and "Yes We Can!" (Barack Obama), it remains to be seen whether a language model can do the same. Our project aims to build a language model that determines the most likely speaker of a given input quote. Morover, our project seeks to investigate how the model discernability changes when trained on three types of data: presidential speeches, part of speech tags for presidental speechs, and politicans' tweets.

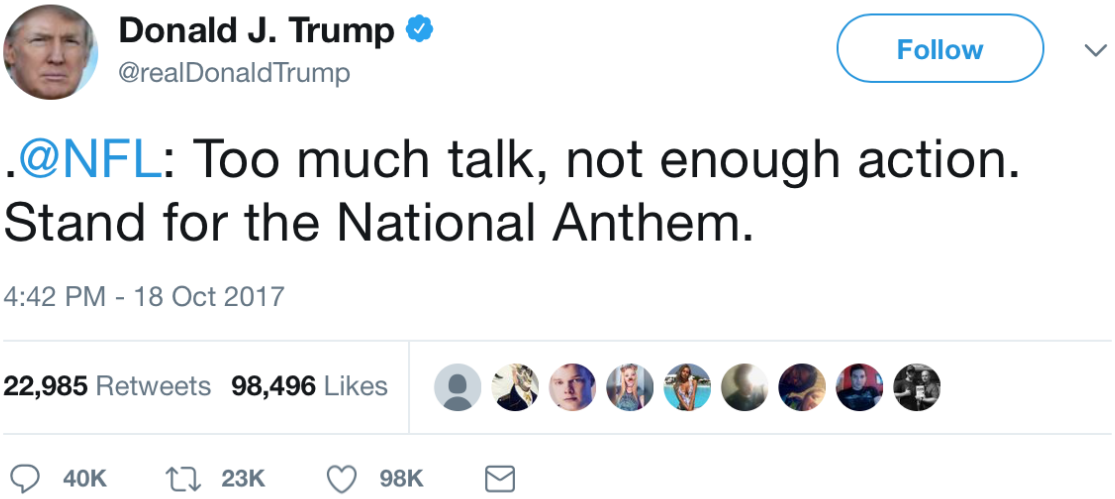
Methodogy

Data Sources

Speeches from each of the 44 presidents and 2016 Democratic Nominee Hillary Clinton were obtained from on open source online corpus compiled by The Grammar Lab. Tweets from Presidents Donald Trump and Barack Obama and various Senators were collected from data made public by FiveThirtyEight.

Data Cleaning

Tweets were parsed and tokenized before using the model. All words were lowercased so that various capitalization wouldn't affect the results. All punctuation (including commas) was removed from tweets and was replaced with beginning and end of sentence speech tags (<s> and </s>). Given that tweets can be quoted (in the form of a hyperlink to said tweet), everything after (and including) the text pattern "http" was removed when such a pattern appeared in a tweet. Additionally, Twitter offers users the option to share or "Retweet" posts that other users have made. Since retweets, indicated in our corpus by RT, aren't necessarily direct words from a given user, all retweets were removed from the training corpus. When all is said and done, this tweet



is represented as follows:

['<s>', u'@nfl', u':', u'too', u'much', u'talk', u'not', u'enough', u'action', '</s>', '<s>', u'stand', u'for', u'the', u'national', u'anthem', '</s>']

Presidential speechs were cleaned in a manner that matched the output of the tweet parser. That is, all punction marks were removed and sentences were marked with appropriate tags. The first two lines of each document was removed as it provided the name and date of each speech. A few speechs were transcripts of interviews, and thus had reporter questions that were not presidential text. Given that these interjections were infrequent and could not be easily removed with regular expressions, we decided to leave them in with the assumption that they would not change any bi-gram or tri-gram probabilities in a significant way.

Fitting the Model

The model we choose to implement is an n-gram model with $n = 2$ or 3. To deal with the issue of unseen counts, we use Good-Turing Smoothing, explained in more detail below. Bigram and Trigram counts are computed from the various training corpuses. We will use three different types of data to train the model in order to see if any type of data works best. They are as follows:

1. Text from presidential speech corpus.
2. Part of speech tags from presidential speech corpus.
3. Tweet text from Obama, Trump and several U.S. senators.

Since the corpus of presidential speeches does not come with POS tags, we use a Hidden Markov Model to compute the most likely tag sequence for each sentence. The HMM is trained on data from the Brown Corpus built into Python's Natural Language Toolkit library.

Good Turing Smoothing

WILL's SECTION HERE

Testing the Model

For each of the three types of training files, we have compiled a list of 10 quotes, each from a different speaker. The 10 possible speakers are then used as training files for the model. The perpexity of each quote for a given speaker is calculated as follows, using addition of logarithms to account for small probabilities.

$$Perplexity = -\frac{1}{N}\sum_{i=1}^{N} log_2(p(w_i|w_{i-1}))$$

Computing perplxity in trigam case is simiar, with the associated trigam probability replacing the bigram probability in the formula above. After computing the perplxity of each sentence for all possibile speakers, we attribute to each quote the speaker that yields the lowest perplexity for the given quote. That is, our model predicts the speaker S of a quote Q as follows:

$$\hat{S} = \arg\min_{s} Perplexity(Q|s)$$

Results

Presidential Speech Corpus

Part of Speech Tags

Tweet Data

Discussion