Courtney Solano
Erin Ruby
Sofie Mehrotra
@officialjaden22
CSCI 4802

Jaden Smith Tweet Bot Project

Summary

Hello And Welcome To The Jaden Smith Tweet Bot Write Up Final Report, By Courtney Solano, Erin Ruby, And Sofie Mehrotra. The goal of this project was to create a tweet bot that imitated Jaden Smith's tweets. His tweets usually follow a very specific pattern where the first letter of each word is capitalized. His tweets also do not make much sense, which makes our project a doable task. It would be difficult to generate logical tweets using any training set, which makes Jaden Smith perfect for this task since his tweets follow a set pattern and make very little semantic sense. We evaluated the bot by the number of likes, retweets and replies each of the bot's tweets got.

There are three different models we used to generate tweets. The first model was a Mad Libs type approach using part of speech tagging to create a template for a tweet, then filling in the template with words from the corpus from the same part of speech. The second model was a bigram model, where each word of the tweet was generated randomly, based on probabilities, from a list of word pairs. The tweets started from a beginning of sentence tag and words were picked until an end of sentence tag was found, or we reached the 240 character limit of tweets. The third model was coming up with our own funny Jaden tweets and tweeting them. This model acted as a baseline to compare our other two models with things that humans think are actually good Jaden tweets.

Process

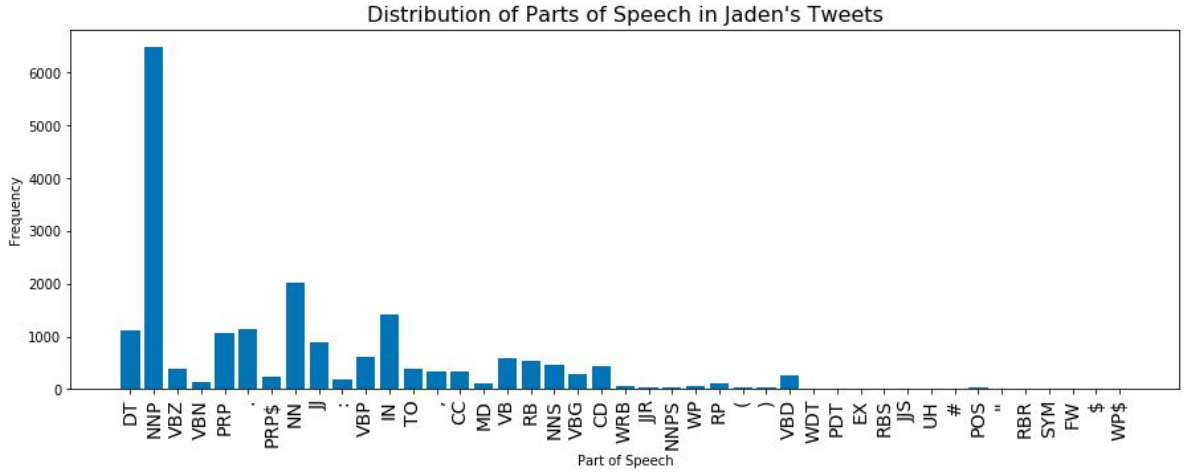
For both the Mad Libs and the Bigram approach, we began the process by collecting data. We began by downloading a large number (2,600) of Jaden Smith's tweets into a json feed, which we then converted to an excel spreadsheet. Then, we read this data into a Jupyter Notebook which we have residing in a public github repository (https://github.com/courtsolano/jadensmith). We noticed Jaden Smith frequently tweets with links and photos. These scenarios are instances we do not intend to replicate so we removed them from our data set. We decided to keep hashtags, mentions to other users, and emojis in order to make our Jaden Smith tweets more realistic. Our cleaned dataset contains tweets that are text and emojis only by Jaden Smith which we will use in teaching our tweet generator.

Creating the twitter account presented some challenges. First, we had to make a Twitter developer account and register an application. We had to tell Twitter what we were planning on making and using the application for. Once it was created, we followed the *Twython* steps in generating keys to securely connect to the application in the jupyter notebook, and creating a Twitter instance that we could use to tweet from inside the code.

We ran into an issue after tweeting about 50 tweets. The account was banned for suspicious behavior, which we suspected was from tweeting nonsense. We had to verify that we were human, and have Twitter send a secure code to one of our cell phones in order to get the account back. Since this roadblock, we have not had this happen again yet, but we imagine it will become an issue again in the future, since this really is a bot tweeting and not a human. We apologize for lying to Twitter. In our defense, this is for educational purposes.

Templating Approach

In order to successfully create the templating/Mad Libs model to generating Jaden Smith tweets, we downloaded the *Natural Language Toolkit (*`NLTK`*)* to help with part of speech tagging tweets. From this library, we downloaded a `tweet_tokenizer` to tokenize each character of each tweet, including mentions to other users and hashtags. The function, `pos_tag`, is a part of speech tagger that tags each token with its respective part of speech. This is where Jaden's tweet style became somewhat problematic. Because he likes to capitalize most of the words in his tweets, the `pos_tag` function was tagging almost everything as proper nouns instead of regular nouns.



The most frequent part of speech was the NNP, or the proper nouns, followed by NN, or nouns. We considered converting everything to lowercase, but then we would have the problem of actual proper nouns not being tagged properly. After researching, we decided to just keep them tagged as proper nouns, as it did not seem to impact the output of the tweetbot very much. We created a dictionary where the keys are the part of speech, and the value corresponding to each key is a list of words that were tagged with that part of speech. We then created a list of our tweet data that contained a tuple for each word in each tweet, with the first entry as the the original word and the second entry as the respective part of speech. This list of lists of tuples is the template for each tweet that we will use to generate new tweets.

Once we were satisfied with the data structures we used to hold the data, we wrote a function called `generate_tweet`. This function picks a random tweet from the list of skeleton tweets and fills each part of speech slot with a random word from the dictionary at the part of speech key associated. The tweet is assembled and returned. We then used the Twython instance created previously to tweet out the tweet to the Jaden Smith Twitter bot using the `update_status` function.

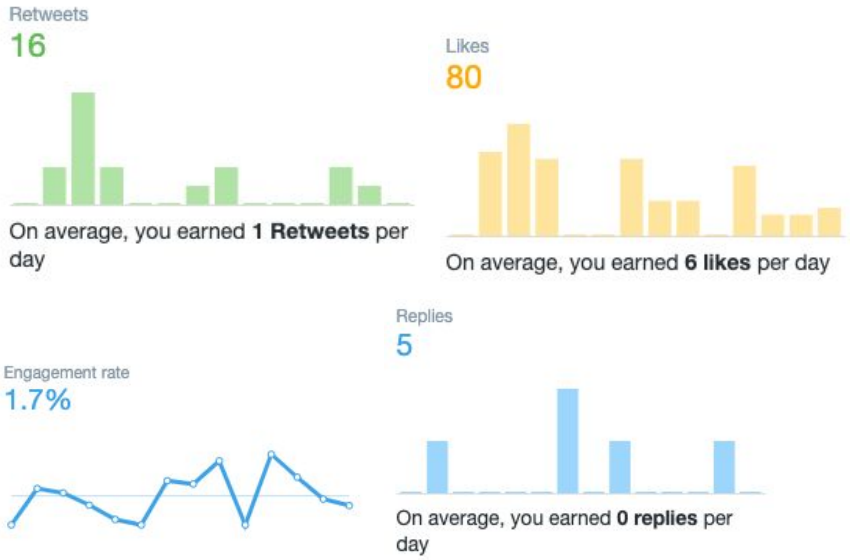
Bigram Approach

Another approach we used to generate Jaden Smith tweets was a bigram model. A bigram is a sequence of two adjacent words. The bigram model takes pairs of words and calculates the probability that the two words appear next to each other, given the training set. The bigrams and associated probabilities are stored together. After the probabilities are calculated, we generated random sentences (tweets) by using *Shannon's Method*.

To create this bigram model, we began by getting all of the words and their counts by using `Counter from collections,` which returns a bag of words, with every word and its count in the training set. We then could use that to clean the training set by setting all the words that only occur once to be `<unk>`. The training data is now a list of words, with `<unk>` in place of the words that only occurred once. Then, we passed this list into a function that creates all of the possible bigrams in our training set, including words that are not next to each other and sets the count of every bigram combination to 1. This process is called *Add-1 Smoothing* and is necessary to account for cases that have a zero probability. Next, we passed the dictionary with all the bigram combinations and the list of unique words to a function that counts which bigrams actually occur in the training set. We use a loop to go through the unique words and increment the count in the dictionary for `uniqueWords[i], uniqueWords[i+1]`. We now have a dictionary that contains all of the bigrams, and their respective counts. Next, we found the probabilities of each bigram. The probability of a bigram is equal to the count of the bigram divided by the count of the first word plus the length of the unique vocabulary. We used this formula to calculate the probabilities of each bigram. Finally, we used *Shannon's Method* to generate the random sentences (tweets). This method begins with a sentence starter marker, `<s>` and a random number between 0 and 1. A random bigram that starts with the end of the previous bigram, as well as has a probability equal to the randomly generated number, is added to the sentence. It goes until an end of sentence marker `</s>` is found. And now, we have a tweet generated by the model. We then used the `Twython` instance's `update_status` function to tweet the generated tweet out to the world to see on Twitter.

Results

There is a function that we were able to turn on called *Tweet Analytics,* where Twitter provides data about how much visibility and interaction your tweets get. "Your Tweets earned 9.7K impressions over this 14 day period". Our top 7 tweets with the most impressions (seen by the most twitter users) were all tweeted from the bot, and had 240+ impressions. However, the tweet with the most engagements (any click in the tweet, retweets, replies, follows, likes) was a manually tweeted one. Our highest engagement rate in one day was only 4%, which shows that people who see our tweets are usually not interacting with them. The engagements did not seem to have any sort of trend, which we interpret as a sign that the bigram model did not improve the tweet bot's performance since we implemented that halfway through the training period. Obviously, the tweets that got retweeted by users more often were viewed by more people, as we only have seven followers. Therefore, the next step would be to find out how to tweak our models so we are retweeted more often and we can begin to gain more interactions and followers. Our twitter bot account can be found at https://twitter.com/officialjaden22.



Model Comparison

There was not a big difference between the interactions computer generated model tweets. One minor difference is that with the bigram model, we had to truncate some tweets if they went over the character limit of 240 characters per tweet. This was not a big issue with the template model, because most of the tweets were short, and words were replacing other words instead of being generated in sequential order. It was clear that the tweets that had the most interactions were the ones we came up with on our own. This could be due to the fact that people thought the tweet was randomly generated, which makes it more humorous. Clearly the tweet generating models have a long way to go before they can be as realistic as humans.

Goals

I would say that we accomplished our goal of learning how to use machine learning to create new content, and in which ways were the most successful in creating tweets. It is clear to us that some tweets were more believable as Jaden tweets than others, just because some of the time the models were lucky enough to produce logical sentences. Also, we were able to accomplish the goal of becoming friends with Hunter because we talked at the Downer for like an hour that one time. Also we know his favorite color is seafoam green, so if that doesn't define friendship, I don't know what does. Would love to grab a drink with him again if possible. Anywho, I don't think our model is quite to the point where we can fool people into thinking that our tweetbot is the real Jaden Smith. Ideally, we would make a sort of metric for accepting/rejecting generated tweets so that only the realistic ones would be tweeted out. Some of them were very Jaden-y, such as one that just said "Vibes." (Sofie's personal favorite), but some of the tweets were absolute nonsense, mentioning random people. Also, not all of the words were capitalized, because he didn't capitalize all of the words in his tweets. In the future, it would be good to make the first letter of every word capital to be more accurate to the template we tried to follow.