# GPTweet Report

**Jordan Nazemi**
University of Loyola Chicago
jnazemi@luc.ed

## 1 Abstract

GPTweet is a means of collecting data using a defined search term from Twitter, sorting based on sentiment analysis, and compiling the sorted data into a corpus that is used to create a model that can generate original human-sounding Tweets. The example model I defined searches for only negative-sentiment tweets concerning American presidential candidate Joe Biden. The reason being that, as the name GPTweet suggests, this paper is meant to show how easy it is to create Twitter "shill" accounts which are used en-masse to influence public opinion. The main goal of the system is to provide human-sounding tweets that are almost impossible to differentiate from real ones.

## 2 Introduction

A report by The Atlantic found that during the 2016 election nearly a third of pro-Trump Twitter accounts and a fifth of Hillary ones were automated [1]. These accounts, many run by foreign entities, were pandemic in any Twitter feed related to the elections. Now, with the looming 2020 election, most users have forgotten about this online "fixing." However, new technology like the recently released GPT-2, a powerful unsupervised Transformer language model, has made generating fake Tweets all the easier [2]. The model uses Tensorflow as its base which is a very common open-source library used when programming machine learning models.

GPT-2 was developed by OpenAI, a research organization based in San Francisco. They've largely been considered the primary competition to Google's own Deepmind project in neural networks. OpenAI has worked on a number of big projects, one of the most notable being GPT-2 itself. Originally, only the smallest model was available to the public due to concerns its efficacy might have in the realm of fake news. As of yet, there hasn't been any major news of GPT-2's usage for criminal activity but I believe it's only a matter of time. I'm trying in this paper to use this powerful model to generate unique human-sounding text that would not look out of place in a normal Twitter feed.

# 3 Background

The following examples are roughly similar projects related to language generation. Although their purpose is not to deceive like GPTweet, they likewise imitate human-sounding sentences.

## 3.1 Mexica

Mexica is a story generation system that takes characters and uses a series of predefined "story actions" to develop a narrative [3]. Mexica is more systematic in its approach but has a far more limited creative space. It takes the narrative location in the story for each character, then decides from a list of actions what the character will do and to whom. For example, there would be a check to make sure the main character is alive and in proximity to the villain before they are capable of fighting. The big limiter in Mexica is this defined list of actions. There is no room to expand from it, and because of it the sentences in the story often come across as stiff or unconnected.

The system was originally my inspiration in class for focusing on language generation, but besides from that it's a very different system in purpose and technique from the blackbox that is GPTweet's model.

## 3.2 Metaphor Magnet

Metaphor Magnet is a system underlying a Twitter bot that generates random metaphors, rhymes, and wordplay [4]. It breaks down words into core features and uses them in a relational database. For example, a wheel would likely be listed with "@wheel:move" as well as a dozen other features. It then uses these related features to generate. This built-in feature transformation system, titled Flux Capacitor, is exceptionally well-implemented and allows for a wide range of possible outputs, connections, and comparisons. It's output being a Twitter bot makes it even more similar to the system I've developed.

The Metaphor Magnet is a possible inspiration in how improvements could be made to GPTweet using a potentially similar transformation system. It also uses a series of easy-ready formats for witty outputs that could be implemented in my design.

# 4 Methodology and design

## 4.1 Pulling Data

The first requirement was to design a system for pulling data en masse from Twitter. The best method for this I found was using TweePy (a custom Twitter web scraping library) and an official authorization code to access Twitter API [5]. After registering my application with

Twitter’s development site I set up my TweePy script. The Twitter API  places a cap on requests between 50 and 100 thousand Tweets per 10 minutes. This posed a challenge as restarting the requests after the rest time often led to pulling duplicate Tweets as it would reset the timeline. By accessing and storing Tweet ID’s I was able to circumvent this issue and return to the same place every time requests would surpass its limits.

## 4.2 Analyzing Sentiment

In order to influence the model to generate primarily NEGATIVE Tweets about the search term (fake political Twitter accounts always have an ideological position), the corpus must be filtered by only negative Tweets. This is done via sentiment analysis of which dozens of models and libraries exist in Python. There were three main options: IBM Watson, Textblob, and NLTK Vader. I went and tested all three with varying amounts of success. The most effective model was IBM Watson which was capable of successfully identifying sentiment in sentences often rated as Neutral or Slightly Positive by NLTK and Textblob. It was able to recognize when a tweet was pro-Biden but formed as a negative-sounding attack on another candidate (this often came up as Strongly Negative in the other systems). Despite this, my computer was incapable of running Watson efficiently and I couldn’t get it to run on a Google collab server at all. This ended up proving Watson to be too cumbersome an option for the massive amounts of Tweets TweePy was pulling in. Moving to Textblob and Vader, I was unable to really tell the effective differences between these two libraries, however, after having read a paper on Vader's usage in analysis sentiment from social media I chose it for my project. [6]

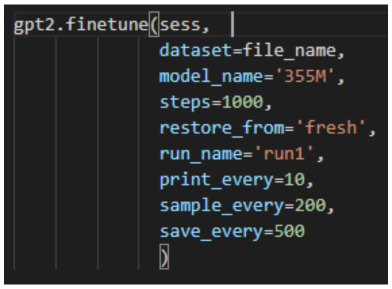
## 4.3 Creating a Model

I used GPT-2 and the GPT-2-Simple API to form a model based on the corpus previously generated and filtered. Several versions of the GPT-2 model exist but for the sake of time, I used the medium-sized model (355M model). Although the code remains relatively unchanged, I used Google collab servers to expedite the process. As such, the default model used in the Model Builder packaged with my report uses the more lightweight 124M model. I found that anywhere between a thousand and two thousand steps is enough to form an effective model, any more steps and it tends to ‘over-train’ itself and pull TOO closely from the corpus.

## 4.4 Generating Results

Once a model has been trained it is rather easy to generate examples. All that’s required is a set length, the amount of “batches” to generate, and an optional word-term to start it off. Once generated the results need to be looked over by a user and hand-picked for use.

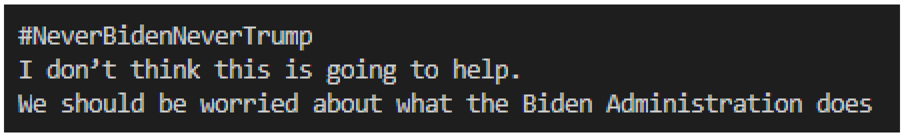
## 5 Results



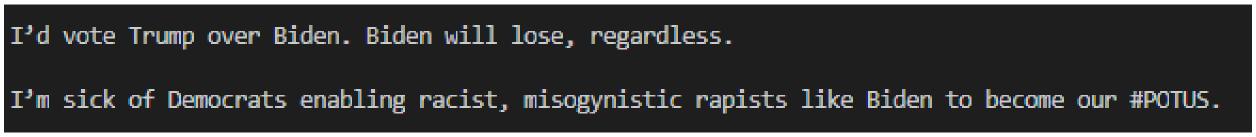
```
gpt2.finetune(sess,
              dataset=file_name,
              model_name='355M',
              steps=1000,
              restore_from='fresh',
              run_name='run1',
              print_every=10,
              sample_every=200,
              save_every=500
              )
```

These were the settings used to generate the model using GPT-2's 'finetune' method. The corpus used was compiled from searching 1 million 'Biden' associated Tweets and filtering by negative sentiment. This ended up being over one hundred thousand tweets that took up nearly half a million lines in a plain text document. Below are example results taken from that model.

The examples below are uniquely generated. Having searched the corpus none of these show up in their complete form already. At a glance, we can see they bandy out commonly used talking points against Biden and the appropriate usage of related hashtags. Although forming the model was very processor-intensive, the generating of examples is done quickly and easily and comes up with rather convincing tweets. One issue I found is that I was unable to put a hard-cap on the size of generated examples, making it difficult to keep them believably within the 280 Twitter character cap.

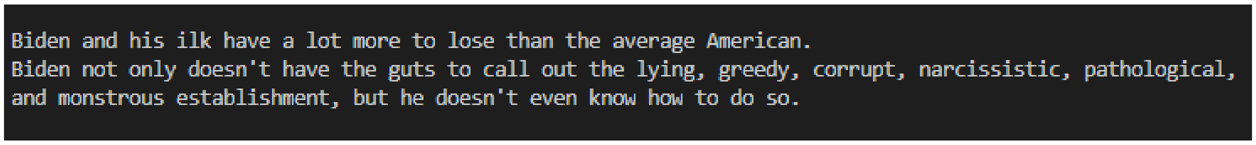


```
#NeverBidenNeverTrump
I don't think this is going to help.
We should be worried about what the Biden Administration does
```



```
I'd vote Trump over Biden. Biden will lose, regardless.

I'm sick of Democrats enabling racist, misogynistic rapists like Biden to become our #POTUS.
```



```
Biden and his ilk have a lot more to lose than the average American.
Biden not only doesn't have the guts to call out the lying, greedy, corrupt, narcissistic, pathological,
and monstrous establishment, but he doesn't even know how to do so.
```

## 6 Evaluation

### 6.1 Quality of Samples

Looking at the output from the model many of the generated examples are quality enough to be used. However, there are often artifacts of broken English and nonsensical characters in the output, perhaps confused by the multilingual aspect of Twitter. As it stands I couldn't think of an effective way to evaluate the quality of the generated examples. In this form, the model requires human interaction in picking and choosing which results are good enough to be shown.

However, this is conjecture since due to my agreement with Twitter in using their API I am disallowed from using my model to post actual tweets.

## 6.2 Coltons Creative Tripod

A good means of determining whether a system is creative or not is to use Colton's Creative Tripod criteria. According to Colton, the qualification of whether a program can be considered creative or not has the primary aim of "the behavior of the software cannot be predicted by the programmer." A creative program exhibits behavior that is skillful, appreciative, and imaginative.

### 6.21 Skillful

This is the ability of the software to understand and have mastery over the creative space it works within. In my opinion, the GPT-2 engine and the model I built with it have effective control over its creative space. It's able to form nearly wholly unique sentences that make logical relational sense. For example, the model has noted from its corpus that Biden is often accused of sexual harassment in attacks against him and now often generates unique output referring to this. It understands the actions, thoughts, and intentions often associated with Biden tweets and is able to recreate them often seamlessly

### 6.22 Imagination

The potential for a system to innovate its methods. This one is arguable for my model. Since GPT-2 (and the Tensorflow library it's built on) are a Blackbox, it's impossible to tell just how much the model is innovating its approach. Fundamentally, however, the purpose of the model is not to generate unique and interesting output, but instead to mimic a very specific type of human-sounding tweet. Within that realm, there is not much purpose for innovative approaches.

### 6.23 Appreciation

The ability of a system to evaluate its output. This is the weakest aspect of my model. As it stands, I have no means of automatically evaluating the quality of its output. The model has no understanding of its own efficacy and cannot improve itself.

# Conclusions

---

In this project, I collected data, formed an appropriate corpus, and trained a model capable of generating anti-Biden tweet messages. It represents a creative-like system that uses a BlackBox to make effective imitations within its creative space. I believe the system, like most neural models, has a lot of room for tweaks in both its training and corpus building that could improve its efficacy.

I say creative-like since evaluating it using Coltons Creative Tripod shows that it fails to hold up two of the three pillars of a creative system: imagination and appreciation. Despite this, I'd say the system succeeds in its ability to make unique outputs.

An important addition to this project would be the ability of the network to evaluate the normalcy of its own tweets. This is, however, quite a high-level problem that's being actively faced by machine learning research.

A huge amount of credit goes out to the developers at OpenAI and their fantastic GPT-2 model, this project would have been far more difficult without it.

## Bibliography

1. https://www.theatlantic.com/technology/archive/2016/11/election-bots/506072/
2. https://openai.com/blog/better-language-models/
3. https://www.researchgate.net/publication/220080099_MEXICA_A_computer_model_of_a_cognitive_account_of_creative_writing
4. https://dl.acm.org/doi/10.5555/2390470.2390472
5. https://www.tweepy.org/
6. http://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf