"Deep Faking" Political Twitter using Transfer learning and GPT-2

Ryan Ressmeyer
Department of Electrical Engineering
Stanford University
ryanress@stanford.edu

Sam Masling
Department of Computer Science
Stanford University
smasling@stanford.edu

Madeline Liao
Department of Computer Science
Stanford University
mmliao@stanford.edu

Abstract

In the modern political climate, Twitter has become one of, if not the most impactful mediums for both current and aspiring politicians to communicate with their constituents.

Our group attempted to replicate a variety of politicians' twitter accounts using the recently released GPT-2 model from OpenAI and transfer learning. We compared our results to former state-of-the-art LSTM recurrent models to illustrate the advantages of transfer learning. Additionally, we used GPT-2 to extrapolate from a given context, allowing for robust tweet mapping from one user to another.

1 Introduction

In recent election years, Twitter has become an increasingly important communication platform for American politicians. Our goal was to see if we could use state-of-the-art NLP techniques to not only recognize the different style and contents of tweets written by different politicians, but to actually generate similar tweets. This idea stemmed from two popular fields of study within NLP: Twitter sentiment analysis, and zero-shot text generation. By utilizing transfer learning and building upon OpenAI's GPT-2 text generation model, we sought to train a model for the domain-specific task of generating tweets based on a given account.

OpenAI's GPT-2 is a transformer-based language model trained with the goal of predicting the next word given all previous words of some text [8]. Upon being given some prompt, the model has been shown to be able to generate seemingly human-authored text in response to the prompt that makes sense both grammatically and semantically. We wanted to extrapolate this model to generate tweets given a dataset of tweets to train on, rather than responding to a given prompt. Specifically, we wanted to be able to extract a topic from a given tweet and train on generating tweets about said topic.

We scraped about 2000 tweets from each account we chose to "deep fake," and for each tweet, we used RAKE to extract a list of keywords, and then for each keyword, we used GloVe to generate a set of similar words that were then appended to a "history" for the tweet [9]. Finally, based on the dataset of tweets with their corresponding histories, we were able to generate new tweets via GPT-2.

2 Related Work

2.1 Recurrent Neural Networks and LSTM for Text Generation

Recurrent neural networks, specifically, long short-term memory networks, are commonly used for text-generation tasks. Sequence to sequence encoder/decoder models, such as that provided in by Sutskever et. al. (2014) [11], utilize LSTM to map sequences of text to a newly generated sequence of text, which has been shown to be effective for translation and dialogue texts [10]. LSTMs have also been used in previous works on tweet generation; the most similar existing work to our project is that of MIT postdoc Bradley Hayes: @DeepDrumpf, a Twitter bot that posts Trump-like tweets. Hayes's model uses a recurrent neural network that generates tweets one character at a time. [4].

2.2 The Transformer Model

Recurrent neural networks are computationally expensive, and training cannot be parallelized due to its sequential nature. The transformer model alleviates this problem by neither using recurrent nor convolutional neural networks and only using attention mechanisms, which has allowed for text-generation models that are much less computationally expensive [12]. OpenAI recently trained a state-of-the-art transformer based language model on over 8 million online documents [8]. These 40 GBs of text, coined WebText, were curated by scraping the web with an emphasis on document quality. The model OpenAI created, GPT2, expanded on their original with slight layer normalization tweaks and an expanded vocabulary and context size. Without fine-tuning, it broke previous accuracy records for seven out of eight datasets. Indeed, it outperformed these previous records without ever actually training on the datasets themselves.

OpenAI's team conclude their paper with a recognition that their work has purely examined GPT-2's zero-shot performance. That is, how it has been able to generalize its unsupervised training to new problems. However, they speculate that GPT-2's potential far exceeds these metrics as it can easily be fine-tuned to tasks using transfer learning. For this reason, OpenAI only released the 345M parameter version of their full 1.5B parameter GPT-2. However, researches have still began to use these models to extend GPT-2 to domain specific uses. Some results include Lee, Jieh-Sheng, and Jieh Hsiang's (2019) adaptation of GPT-2 to generate patent claims, and Budzianowski, Paweł, and Ivan Vulic's (2019) task oriented dialogue bots. This paper hopes to add to the growing understanding of GPT-2's capabilities by fine-tuning it to generate tweets in the style of a specific user.

To achieve these results using transfer learning, Golovanov, Sergey, et al.(2019) specifies two means of adapting a zero-shot, unsupervised transformer model to a task specific language model. So called single-input training concatenates the context for language generation to training examples such that only a transformer-based decoder is needed to begin generation. Conversely, multi-input training uses an encoder-decoder structure to load context into a transformer model. Within the realm of this paper, we used single-input training examples due to their widespread use in recent GPT based transfer learning papers [1][13].

2.3 Model Selection for Tweet Generation and "Deep-Faking"

While LSTMs are commonly used for text-generation problems similar to the one we attempt to solve, a few factors led us to our ultimate decision to use transfer learning with the GPT-2 transformer model. Firstly, given the vast pre-training GPT-2 has courtesy of OpenAI, a transfer-based approach would retain the inherent knowledge of language that GPT-2 contains. This is important given the limited number of tweets we were able to attain from each user on twitter. As we found with our baseline LSTM, there was be no inherent knowledge of the english language, and thus outputs were random and lacked a coherent sentence structure. Secondly, given our focus on mimicking the content and style of specific politicians, we chose a transformer based model to allow our model to "focus" on learning each politician's twitter personality given a specific context rather than simply learning how to complete a sentence (as was the case with LSTM). Additionally, a disadvantage of recurrent neural networks is that the less recent context is eventually "forgotten" by the model when generating the next word. With transformer models, a sentence's topic is never forgotten, allowing for robust tweet generation. Thus, we decided to move forward with GPT-2 using transfer learning.

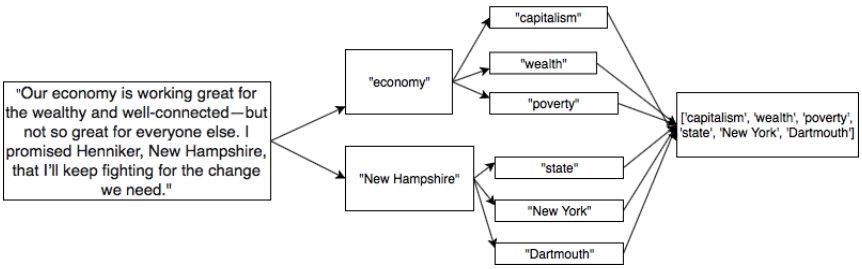

Figure 1: Context extraction process.

3 Dataset and Features

To generate our dataset, we used Twitter's official API to gather the most recent tweets from numerous politicians. Twitter's API allows you to scrape up to 3000 of a given user's most recent tweets. These tweets, however, will include retweets, which we had to filter out given that we wanted to capture the specific style of each user, which is not shown in a retweet. This left us with around 2200 original tweets for each user we looked at. We then processed each tweet, filtering out images and links which left us with plain text for each entry. While links and images are often an important part of a twitter user's persona, we chose to filter these out as the goal of this project was to capture the grammar and word usage of a user using GPT-2, which was not built to deal with images and links.

Along with the filtering, we also used keywords to generate context or a "history" for each tweet (the reasoning behind this will be addressed in our introduction to transformers later in the paper). Using RAKE (Rapid Automatic Keyword Extraction), we extracted monogram and bigram keywords from each tweet. From there we used GloVe vectors to find the 10 most similar words to each key word, and used the aggregation of these similar words as the context for each tweet.

Thus, we see that for the tweet in Figure 1, a simplified context could look like 'capitalism wealth poverty state New York Dartmouth'. We used these contexts to help our model develop an understanding of what tweets on similar topics look like. This also allowed our model to generate tweets on topics that were tangential to the subjects of the tweets in our dataset, instead of only being able to generate tweets directly from the content matter of the dataset.

4 Methods

4.1 Training Example Formatting

Given the RAKE then GloVe extrapolated context for each tweet, our training algorithm then generated training and validation examples. Following the single-input method described in Golovanov, Sergey, et al.[2019], our model was trained using a tweet's context concatenated with the actual tweet. To specify which user a tweet came from, we add the tweeter's name to the beginning of the input sequence. In addition, we use a sentence segment embedding layer to denote which parts of the input were the tweeter's name, context for the tweet, and tweet itself. Finally, since GPT-2 is not a recurrent neural network, a positional embedding layer is added to give the model a concrete sense of position for each word.

4.2 Multi-task Loss

To achieve our goal of fine-tuning GPT-2 to generate tweets in a similar style to a given twitter user, we used a multi-task loss defined as a linear combination of *language model loss* and *Multiple Choice Loss*. This combination of loss functions was found in Wolf, Thomas, et al. [2019] to drastically increase dialogue readability and adherence to context. *language model loss* is defined as the cross-entropy loss applied to a softmax of the transformer decoder output with the example tweet

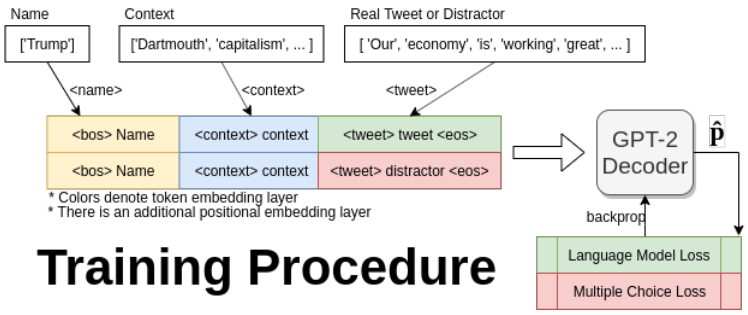

Figure 2: Visualization of the training process. Includes input representation and multiple choice loss

words as the labels. Given the word sequence $S = \{w_1, ..., w_{|s|}\}$, *language model loss* is defined as:

$$l_{lm}(S) = -\sum_{i=1}^{|S|} \left(\log \hat{P}(w_i | w_{i-1}, ..., w_1) \right)$$

As described in Wolf, Thomas, et al. [2019], *Multiple Choice Loss* involves adding "distractor" examples for each true example in the dataset. As previously mentioned, these examples have the context of a previous tweet, but the training tweet itself is randomly selected from other twitter accounts. *Multiple Choice Loss* involves calculating a final hidden layer after the sentence has ended. This final hiden layer, h_l, is used as the input to a linear classifier layer to correctly classify the true example among distractors. During training, this classifier is trained jointly with the transformer model using the loss function:

$$l_{mc}(S) = -((y \log(\sigma(h_l * W_h)) + (1 - y) \log(1 - \sigma(h_l * W_h))$$

Where W_h is learned during training, y is defined as 1 if the example is the true example and 0 otherwise, and σ is the sigmoid function.
At this point that it is worth mentioning that both of these losses, and the multiple choice classifier are automatically generated in our code during training using the Huggingface transformers library for Pytorch. It is open-source and dramatically reduces the barrier to entry into NLP using transformers.

4.3 Transfer Learning Specifics

Our final model was trained on approximately 1400 tweets. It used an equal weighting of language model and multiple choice loss for 4 epochs. It trained with a learning rate of $6.25e^{-5}$ and batch size of 4. Using google cloud computing, the model only took around 40 minutes to train, since the training set was relatively small.

4.4 Tweet Generation

In order to generate tweets, the output of the decoder must be extrapolated into words. GPT-2 takes an input and returns an array of probabilities for each word in its vocabulary corresponding to the probability of that each word follows the input. To generate text, we simply take one of these words and append it to the input and rerun the model. However, choosing which word to append given the probability distribution is difficult. The baseline is to use *Greedy Decoding*, which takes the highest probability word. Interestingly, this has been shown to not generate lifelike sentences. In our model, top k filtering was used to generate tweets. This involves always sampling from the top k number of words from all word probabilities. This addition of randomness among the top probabilities was found by Fan, Angela, Mike Lewis, and Yann Dauphin. (2018) to lead to more variance in sentence structure as well as longer sentences. In this case, $k = 4$ was used.

5 Experiments/Results

For our baseline, we trained a Keras 2-layer LSTM on tweets by Elizabeth Warren and simply recorded the outputted "tweets." For our GPT-2 based model, the procedure was slightly more

complicated. Our code is available at https://www.github.com/JumboJoll/CS229_Final_Project. After augmenting the data as described in section 3, we trained our model on Donald Trump's account specifically. In order to generate output, we then fed in contexts from other accounts. Below, we include a table of results of both our baseline LSTM as well as the GPT-2 model using contexts from Elizabeth Warren tweets as input for our Trump-trained model.

Our LSTM produced results that were contextually relevant, however the actual tweets themselves were not coherent English. Some examples were:

"care banks rake ed workers wants know would high help us free government need past students"
"wall bill right federal week gay bill working federal work cares wear health admin working country"

We saw drastically improved output from our Transformer model, however:

Original Warren Tweet	Generated Response
"A year ago, the Saudi govt murdered journalist Jamal Khashoggi. There's been no punishment for those who ordered this crime; the Trump admin ignores Congress's demands for accountability. We honor his memory by fighting for justice; human rights."	"I want to be clear that I am not endorsing the use of the military or other unlawful methods to kill innocent people. I don't support the use of force or torture."
"Daniel Collins argued in a brief submitted in the Hobby Lobby case that companies should be able to deny contraceptive coverage to employees. Now @SenateGOP and @realDonaldTrump want him on the Ninth Circuit."	"The Democrats are trying to help them, and they are not doing it!"
"My office found that a sanctioned Russian company with Putin ties received a tariff exemption; most exemptions go to foreign-owned companies. The tariff exclusion process is one more way the Trump admin works for big, well-connected; in many cases, foreign-owned companies."	"The House and the Senate are going to repeal the President's Executive Order, and it will be done with a bipartisan majority. This is a very good thing for American consumers. It will protect our jobs and our economy."

Qualitatively, we notice that our generated Trump tweets 1) reflect the content from the Elizabeth Warren tweet, and 2) reflect both Trump's tweet style as well as his political views.

We also were able to calculate the accuracy of the model as well as the loss at each epoch.

	Accuracy	Loss
Epoch 0	0.593058	1.141
Epoch 1	0.949469	0.813
Epoch 2	0.978723	0.540
Epoch 3	0.984943	0.483
Epoch 4	0.992021	0.454

6 Conclusion/Future Work

Using transfer learning to convert GPT-2 to our domain specific task proved to be more successful using an LSTM. Using contexts from Elizabeth Warren's tweets as the input for a model trained on Trump's tweets, we were able to generate coherent Trump-style tweets about the content from the Warren tweets. Granted, we did not spend a significant amount of time fine-tuning our LSTM; however, for our goal of generating persona-specific tweets, transfer learning on top of a transformer model proved more worthwhile than tuning hyperparameters to ensure that our LSTM learned both coherent English in addition to a persona.

We plan to build upon our results by implementing a more robust evaluation method for our tweets. Given the subjective nature of the quality of text generation (and even more subjective nature of tweet generation, since tweets are often written concisely and informally), we chose to use human analyses to evaluate our model. However, including quantitative measures of analysis would allow us more insight into how we could further improve our model. BLEU scores are commonly used to analyze translation models, while loss and perplexity were metrics used in comparison of various text generation techniques Kawthekar et. al. (2017) [4]. However, the evaluation of our model proves to be difficult because we are not necessarily comparing it to common english. Instead, Twitter is a highly specialized language domain that will likely require its own metrics. Thus, additional tweaking of current evaluation techniques and more research needs to be done in order to more robustly analyze our tweets.

7 Contributions

Ryan Ressmeyer: GPT-2 training, debugging, and generation.
Madeline Liao: Background research on text generation models and techniques, data processing and augmentation, tweet context generation
Sam Masling: Background research on a variety of text generation models, baseline implementation, data processing, tweet context generation

References

[1] Budzianowski, Paweł, and Ivan Vulić. "Hello, It's GPT-2–How Can I Help You? Towards the Use of Pretrained Language Models for Task-Oriented Dialogue Systems." arXiv preprint arXiv:1907.05774 (2019).

[2] Fan, Angela, Mike Lewis, and Yann Dauphin. "Hierarchical neural story generation." arXiv preprint arXiv:1805.04833 (2018).

[3] Golovanov, Sergey, et al. "Large-scale transfer learning for natural language generation." Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019.

[4] Hayes, Bradley. "DeepDrumpf." *Twitter account*.

[5] Kawthekar, P., Rewari, R., Bhooshan, S. "Evaluating Generative Models for Text Generation." 2017.

[6] Lee, Jieh-Sheng, and Jieh Hsiang. "Patent Claim Generation by Fine-Tuning OpenAI GPT-2." arXiv preprint arXiv:1907.02052 (2019).

[7] Pennington, J., Socher, R., Manning, C.. "GloVe: Global Vectors for Word Representation." 2014.

[8] Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI Blog 1.8 2019.

[9] Rose, S., Engel, D., Cramer, N., Cowley, W. Automatic keyword extraction from individual documents. 2010.

[10] Sundermeyer, M., Schlüter, R., Ney, H. "LSTM Neural Networks for Language Modeling." 2012.

[11] Sutskever, I., Vinyals, O., Le, Q. "Sequence to Sequence Learning with Neural Networks." 2014.

[12] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.

[13] Wolf, Thomas, et al. "Transfertransfo: A transfer learning approach for neural network based conversational agents." arXiv preprint arXiv:1901.08149 2019.