CS230

# Make AI Great Again: Generating Tweets in 'Presidential' Style Using Recurrent Neural Networks

**Stephanie Scott**
Graduate School of Business
Stanford University
smscott@stanford.edu

**Naveen Srivatsa**
Graduate School of Business
Stanford University
nns@stanford.edu

## Abstract

President Donald Trump's Twitter account has been noted for its distinctive writing style. We attempt to build a recurrent neural network that trains on 2,829 @realDonaldTrump tweets in order to generate new tweets in the style of the president. A neural network that trains on words rather than characters and consists of 4 bidirectional LSTM layers generates tweets that closely resemble @realDonaldTrump's tweets. While the addition of layers has minimal impact on the quality of the output, using dropout regularization reduces training time and overfitting noticeably.

## 1 Introduction

President Donald Trump's Twitter account has been noted for its distinctive writing style by linguists and media pundits alike. In this paper, we explore whether neural networks can generate tweets in the style of President Trump's Twitter account.

The input to our algorithm is a dataset of 2,829 recent tweets issued by the @realDonaldTrump Twitter account. We then use a recurrent neural network to output a generated tweet in the style of @realDonaldTrump. After reviewing the literature and describing the dataset further, this paper explores architecture and hyperparameter choices of the neural network.

## 2 Related work

World leaders' language choices are regularly studied by historians and linguists. Globally, scholars have performed textual analyses of presidents, including comparing two Korean presidents' inaugural addresses to examine the links between language and policy [1] and analyzing Czech presidents' New Year speeches before and after the collapse of the Berlin Wall [2]. In the United States, historians have identified that speaking to the public is one of the most significant roles that a president plays and that presidents make use of deliberate linguistic choices (e.g., invoking symbols) in their communications [3]. However, presidential scholars have noted that the linguistic style of current U.S. president, Donald J. Trump, differs significantly from precedent, with increased use of asides, superlatives, and verbal intensifiers. Historian Kristen Kobes Du Mez said of President Trump's linguistic style: "This kind of pushes the limits of linguistic analysis." [4]

For linguistic applications, recurrent neural networks (RNNs) are widely used. In the 1980s, RNNs were first introduced to learn strings of characters. Since then, RNNs have been used to learn patterns that follow a sequence or vary with time. They have been used to forecast electric load, predict financial movements, and track head movement for virtual reality systems [5]. Although in the 1990s,

CS230: Deep Learning, Spring 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

many natural language generation systems involved creating structured messages (e.g., "the next train leaves at 10 am") [6], there was also a certain level of research in n-gram language modeling [7].

In 2003, Bengio et. al. propose using feedforward neural networks for language, using a feature vector with shared parameters across words that feed into a tanh function into a softmax function [8]. However, a deficiency cited by Mikolov et. al. (2010) is that the network must fix the length of the context. Instead, they apply RNNs that can learn long-term dependencies [9]. In 2012, Sundermeyer et. al. use a Long Short-Term Memory (LSTM) model that becomes prominent in future language models [10].

RNNs and LSTMs have been widely used for language tasks since then. Two prominent use cases include machine translation [11] and automated image captioning [12].

## 3 Dataset and Features

The dataset consists of 2,829 tweets issued by the @realDonaldTrump Twitter account between March 4, 2017 and May 31, 2018. In order to capture only tweets representative of President Trump's writing style, retweets by @realDonaldTrump of tweets from other accounts were excluded from the dataset. Replies by @realDonaldTrump were included, as they represent original content issued by the account.

Tweets were extracted using the TwDocs tool. Some pre-processing was required: President Trump uses a significant number of ampersands that were extracted as "&amp" and therefore were manually replaced with the proper character. Tweets were subsequently saved in UTF-8 encoding so as to be properly read by algorithms. Pre-processing and normalization required for training varied by model and will be described in the subsequent section.

## 4 Methods and Experiments

To generate tweets reflective of President Trump's style, two broad model architectures were explored: (1) character-based models and (2) word-based models.

### 4.1 Character-based models

Multiple LSTM-based character models were developed using the `keras` package, based on a model developed by Sagar Jaiswal [13]. First, data is pre-processed, so that the input is a string of 100 characters and the output is the single character that follows the 100 characters. Accordingly, the data is pre-processed by creating a list of input data that consists of sequences of 100 characters from the tweets, creating a list of output data that consists of the single characters that follow the 100-character strings, and converting them both to integers using a dictionary. Data is reshaped and then normalized by dividing each character's integer representation by the total number of unique characters. The output is converted into one-hot vectors.

The base model uses one LSTM layer that contains 256 LSTM units, applies dropout regularization that drops out 20 percent of neurons, and applies a softmax activation layer. Adam optimizer is used, with learning rate of 0.001, beta1 of 0.9, and beta2 of 0.999.

Multiple model architecture variants were tested. One variant experimented with the impact of two LSTM layers, such that the full model architecture consists of LSTM, dropout, LSTM, dropout, and softmax activation. One variant tested the the impact of a single bidirectional LSTM layer instead of a single feedforward LSTM layer. Another variant tested the impact of two bidirectional LSTM layers.

In addition, in one experiment related to hyperparameter tuning, the learning rate in the base model was increased to 0.01 from 0.001.

In all models, the loss function used is categorical cross-entropy, or $-\sum_i y_i log\hat{y}_i$.

Table 1: Summary of model results under varying architectures and parameters

| Model | Description | Loss | Sample output |
|---|---|---|---|
| Character: Base model | 1-layer LSTM with dropout, learning rate = 0.001 | 1.8479 (after 96 epochs) | uic oi the resuic oi the resuic oi the resuic oi the resuic oi the resuic oi the resuic oi the resui |
| Character: Base + Faster learning | 1-layer LSTM with dropout, learning rate = 0.01 | 2.0593 (after 19 epochs) | people of the fake news the searet wery searri the people of the fake news the searet wery searri t |
| Character: Two-layer | 2-layer LSTM with dropout, learning rate = 0.001 | 1.5376 (after 50 epochs) | and the fake news media is a great honor to welcome the united states and the people of the world fo |
| Character: Base + bidirectional | 1-layer bidirectional LSTM with dropout, learning rate = 0.001 | 1.9569 (after 36 epochs) | the sireate niane on the uarl and seaurity oo the sases and secori big teey io the uasling onft to |
| Character: Two-layer + bidirectional | 2-layer bidirectional LSTM, learning rate = 0.001 | 1.3889 (after 26 epochs) | the fake news media is a great state of the fake news media is a disaster |
| Word: Base model (on characters) | 4-layer, 128-cell Bidirectional LSTMs, learning rate = 0.004 | 1.9735 (after 10 epochs) | A thateres', my of ranto the w/ Yomil resut dime the we the secuse, GRIO |
| Word: Base model | 4-layer, 128-cell Bidirectional LSTMs, learning rate = 0.004 | 0.2435 (after 100 epochs) | general john kelly is doing a great job as chief of staff . all americans strong on mexico on this our country vote for against the next administration . if no support ! |
| Word: Base model + dropout | 4-layer, 128-cell Bidirectional LSTMs, learning rate = 0.004, dropout = 0.2 | 0.2577 (after 100 epochs) | steve bannon will be interviewed by @ seanhannity at 9pme on @ foxnews |
| Word: Base model + dropout + 8 layer | 8-layer, 128-cell Bidirectional LSTMs, learning rate = 0.004, dropout = 0.2 | 0.2875 (after 150 epochs) | michael wolff is a total loser who made up stories in order to sell this would go up classified information ! spending ! |

### 4.2 Word-based models

Multiple word based models were developed using the `keras` package, based on the textgenrnn model created by Max Woolf [14]. Input data is a line delimited text file of tweets. The data is pre-processed using `keras` tokenizer which turns each word into an integer. Each integer corresponds to a token in a dictionary (with a 10,000 default embedding size). After the input data is encoded, it is randomized and broken into mini-batches for training.

The base model feeds the embedding into two sequential LSTM layers that contain 128 LSTM units each. The embedding and outputs of both LSTM layers are concatenated and fed into an attention layer that calculates the weighted average of inputs over time and outputs a context vector. Last, a dense layer applies a softmax activation. RMSprop is used with with learning rate of 0.004. As with the character-based model, the loss function used is categorical cross-entropy.

## 5 Results and Discussion

Overall, we find that a word-based model using bidirectional LSTM layers can produce tweets that resemble the tweets of President Trump. We organize our discussion of results into multiple sections.

### 5.1 Character-based model vs. word-based model

A one-layer character-based model does not seem able to produce output that resembles the president's tweets, in either words or syntax. However, a two-layer architecture that trains on characters can

produce words and phrases that resemble President Trump's tweets. For example, the phrase "fake news media" is produced by a two-layer architecture. In addition, some phrases such as "United States" and "people of the world" represent English phrases that the president would use. Although these phrases are consistent with the English language, the output does not represent coherent sentences or tweets.

Word-based models, on the other hand, are able to produce coherent sentences or tweets. Phrases such as "The fake media is becoming more and more powerful" and "Dow, S&P 500, and NASDAQ close to make America great again" represent proper grammar and also are in a style similar to the president's tweets. Therefore, we can conclude that word-based models more quickly learn plausible syntax and common phrases and can better generate tweets representative of the style of the president.

### 5.2 Learning rate tuning

One character-based model experimented with increasing the learning rate from 0.001 to 0.01. Increasing the learning rate resulted in no meaningful decrease in the runtime of each epoch. However, the loss reached a lower point faster with a higher learning rate than with a lower learning rate, but not by a material amount. In the single-layer character models, with a learning rate of 0.01, at 19 epochs the model achieved loss of 2.0593, in comparison to the model with a learning rate of 0.001, which achieved a loss of 2.1320 at 19 epochs.

### 5.3 Dropout

By increasing dropout from 0 to 0.2 in the word-based model, the model converged to a low loss much more quickly (Exhibit 3). Dropout also appears to reduce overfitting in tweets generated. A comparison of tweets generated from the model with dropout to models without dropout showed a wider variety of tweets and fewer overfit phrases in tweets generated from the dropout model.

### 5.4 Bidirectional layers

The replacement of the feedforward layers with bidirectional layers in the two-layer character-based models reduces the number of epochs required to achieve low levels of loss. Over 50 epochs, the two-layer character-based model achieved a loss of 1.5376. Replacing the feedforward layers with bidirectional layers allowed the model to achieve comparable loss in just 16 epochs. However, each epoch took considerably longer to train with the bidirectional layers (1920 seconds per epoch for the bidirectional two-layer model, vs. 889 seconds per epoch for the feedforward two-layer model).

### 5.5 More layers

For the character model, adding a layer was the most effective change from the base model. With an additional layer, the model went from unable to reproduce President Trump's Twitter style to able to reproduce words and phrases. For the word-based model, adding four layers shows little marginal improvement to ultimate loss or generated tweet quality. Further with additional layers, number of epochs and training time needed to achieve a low loss increased significantly.

### 5.6 Exploding gradients

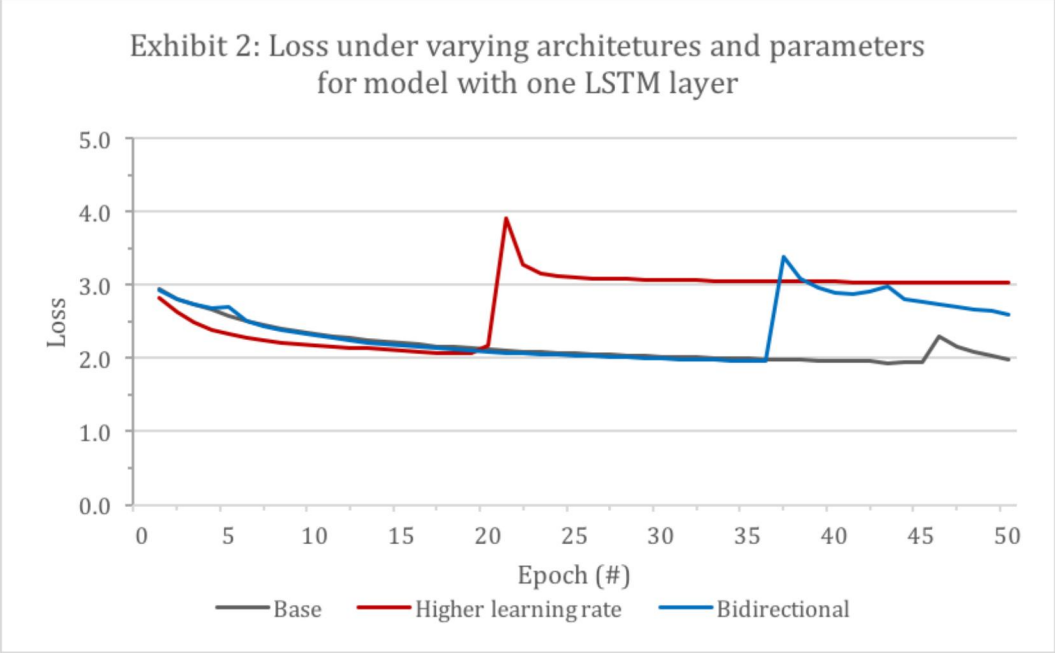
Exhibit 2 suggests that the character-based models with the higher learning rate and the bidirectional layer face the challenge of exploding gradients, in contrast to the base model. As discussed in Pascanu et. al. (2013), exploding and vanishing gradients are common issues with recurrent neural networks, and exploding gradients can be addressed with gradient norm clipping [15]. This was not implemented once the overall superiority of the word-based models was observed.

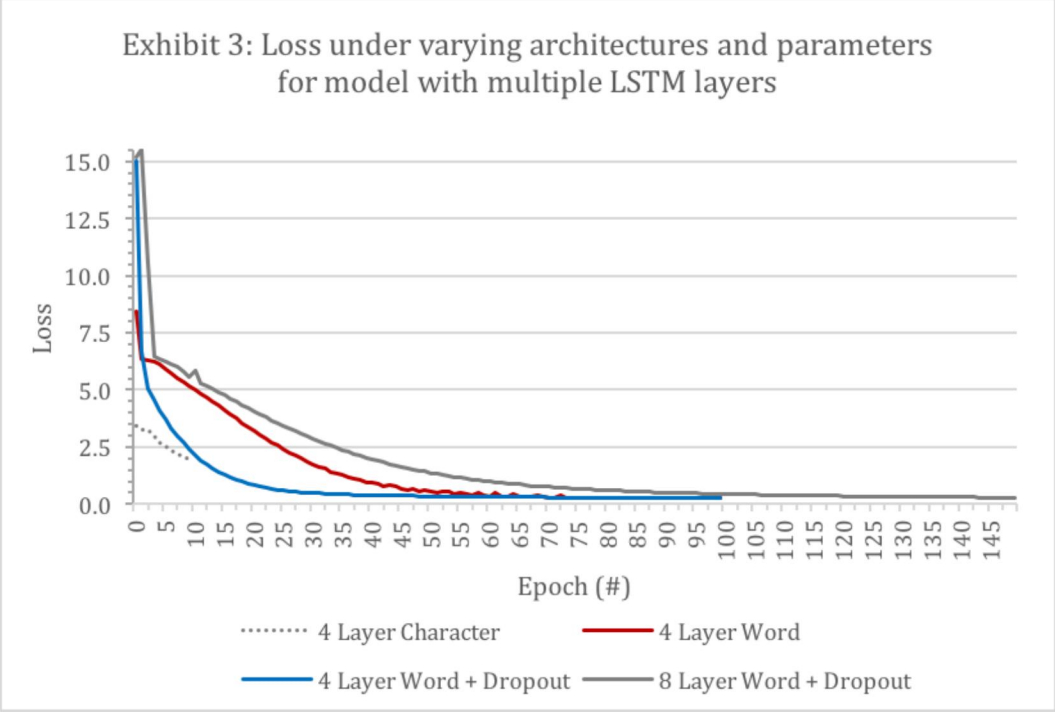
## 6 Conclusion

We find that a neural network that trains on words rather than characters and consists of 4 bidirectional LSTM layers with dropout generates tweets that closely resemble @realDonaldTrump's tweets. A word-based model was superior to a character-based model, given the character-based model faces the twin challenge of learning words and syntax. The addition of a layer greatly enhances the

character-based model, but the addition of four layers has minimal effect on the word-based model. We also see that the character model faces the risk of exploding gradients, as evidenced by the spikes seen in Exhibit 2, while the word-based model faces some risk of overfitting. Future iterations of this model should begin with the word-based model and focus on reducing levels of overfitting.

## 7 Graphs and exhibits



## 8 Contributions

Both team members take equal responsibility for this paper.

## References


[1] Chung, C. J., & Park, H. W. (2010). Textual analysis of a political message: the inaugural addresses of two Korean presidents. Social science information, 49(2), 215-239.

[2] Čech, R. (2014). Language and ideology: Quantitative thematic analysis of New Year speeches given by Czechoslovak and Czech presidents (1949–2011). Quality & Quantity, 48(2), 899-910.

[3] Coe, K. (2007). The Language of Freedom in the American Presidency, 1933-2006. Presidential Studies Quarterly, 37(3), 375-398.

[4] Sedensky, M. (2017, April 26). Trump's speaking style still flummoxes linguists. Associated Press.

[5] Medsker, L. R., & Jain, L. C. (2001). Recurrent neural networks. Design and Applications, 5.

[6] Reiter, E., & Dale, R. (1997). Building applied natural language generation systems. Natural Language Engineering, 3(1), 57-87.

[7] Kneser, R., & Ney, H. (1995, May). Improved backing-off for m-gram language modeling. In Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on (Vol. 1, pp. 181-184). IEEE.

[8] Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. Journal of machine learning research, 3(Feb), 1137-1155.

[9] Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In Eleventh Annual Conference of the International Speech Communication Association.

[10] Sundermeyer, M., Schlüter, R., & Ney, H. (2012). LSTM neural networks for language modeling. In Thirteenth Annual Conference of the International Speech Communication Association.

[11] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[12] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In International Conference on Machine Learning (pp. 2048-2057).

[13] Jaiswal, S. (2017). Keras Recurrent Neural Network with Python. GitHub repository, https://github.com/sagar448/Keras-Recurrent-Neural-Network-Python

[14] Woolf, M. (2017). Textgenrnn. GitHub repository, https://github.com/minimaxir/textgenrnn

[15] Pascanu, R., Mikolov, T., & Bengio, Y. (2013, February). On the difficulty of training recurrent neural networks. In International Conference on Machine Learning (pp. 1310-1318).