Make Deep Learning Great Again: Character-Level RNN Speech Generation in the Style of Donald Trump

Benjamin Sherman
Department of Computer Science
University of California, Santa Cruz
Santa Cruz, CA 95064
bcsherma@ucsc.edu

Zayd Hammoudeh
Department of Computer Science
University of California, Santa Cruz
Santa Cruz, CA 95064
zayd@ucsc.edu

Abstract

A character-level recurrent neural network (RNN) is a statistical language model capable of producing text that superficially resembles a training corpus. The efficacy of these networks in mimicking Shakespeare, `Linux` source code, and other forms of text have already been demonstrated. In this paper, we show that character level RNNs are capable of very believably mimicking the language of President Donald J. Trump after training on a corpus of speech transcriptions. We believe our most significant contributions to the study of character level statistical language models are in sampling methodologies; specifically we propose that introducing dropout during the text-generation phase introduces randomness that leads to more believable text.

1 Introduction

2 Character Level RNNs

A character level RNN is a statistical model of language. It is statistical in the sense that it views the behavior of language probabilistically. To better understand this, allow that my brain uses a statistical language model and that my job is to try to finish all of your sentences. You say “I am going to the grocery-”. I would guess with high probability that you are about to say “store”, though I have to accept that there is a non-zero chance you will say “outlet”- or, perhaps you will stub your toe and say “ouch!” in the middle of your sentence. What a character level RNN does is really no different, except the inferences it learns to make happen on the character level.

Let $\mathcal{C} : V^L \to P$ be a character-level recurrent neural network, such that V is a vocabulary of approximately 100 characters, L is an arbitrary sequence length, and P is the set of all probability distributions over V. More plainly, we can think of a character-level RNN as a function that takes a sequence of L characters and gives us a probability distribution; in particular, it gives the probability of the next character given the previous L characters. As an example of how the network will ideally behave, imagine that you let $p = \mathcal{C}(\texttt{(M,a,k,e, ,A,m,e,r,i,c)})$. If the network is well trained you should find that p predicts `a` as the next character with high probability.

In order to generate meaningful text with a character level RNN you need to give it a seed, similar to the sentence-to-be-finished from the game described earlier. Prompted with the sequence “We will build a g”, the RNN will produce a distribution over the vocabulary. We can sample from this distribution and add the resulting character to the sequence. If the sampled character was ‘r‘, as we would hope, then we now have the sequence “We will build a gr”. We can continue this process for arbitrarily many steps. It is important to note that we can not make networks that accept arbitrarily

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

long sequences as input, so at some point we have to start removing a character from the beginning of the sequence for every character we add to the end of the sequence.

2.1 Impracticality of Word-Level RNNs

It is obvious that generating paragraphs of text one character at a time may yield suboptimal results. As part of the feedback to our initial project proposal, the grader specifically asked why we chose to do a character-level RNN instead of a word-level RNN. Superficially, a word-level RNN superficially has clear advantages including that it is less likely to produce spelling mistakes and also makes decisions at a coarser granularity which may yield superior results. When upon a more detail analysis, it is clear that word-level RNNs are impractical.

First, the current Oxford English dictionary has over 170,000 words. A world-level learner would need a separate output node for each of these words. Such a large output is likely to suffer from underflow and floating point errors that would severely degrade the quality of its predictions. In addition, training such a large network would be prohibitively long and would extend significantly past the short duration for this project. Even Google with its near limitless computation and human resources does not do word-level prediction and instead uses morphemes.

One team in the CMPS242 class chose to use a word-level learner. To address the output-layer size issue mentioned previously, this team reduced the vocabulary to several hundred words. They also only generated phrases of approximately 10 words. Such extreme constraints yield results that significantly underachieve character-level RNNs.

3 Training

3.1 Dataset

Character-level RNNs can be provided any user-supplied text as a seed. While some words are substantially more common than others, it does not change the fact that a character-level RNN must be able to generate meaningful outputs from countless many input seeds. Therefore, to achieve acceptable performance, the training set needs to be especially large.

Although President Trump is credited with being the lead author of over a dozen books [1–13], we also deliberately chose to exclude them from the training set also for two primary reasons. First, many of the books featured co-authors or were entirely ghostwritten [14]. As such, it would be difficult to distinguish Trump's own style from those of his writing partners. In addition, most of Trump's books date back to the late 1980's through the early 2000's. Most of the students in the course had not even been born by the time some of these books were written. Hence, the generated text they may yield may not be meaningful to the class's relatively young audience.

Another possible source of training content are Trump's tweets, but we did not use them in this project for multiple reasons. First, Twitter limits tweets to only 144 characters. As such, tweeters deliberately prune content and commentary to fit within this strict type limit. This leads to extensive use of abbreviations, skipping of words, or hastags (e.g., "#MAGA" for "Make America Great Again") many of which are exclusive to the Twitter platform. For example, one of President Trump's signature Twitter mannerisms is to end a tweet with "Sad!"; however, this is not done in everyday speech even by the president himself. In addition, similar to at least some of Mr. Trump's books, many of his tweets are ghostwritten. It has been reported that at least White House social media director Dan Scavino Jr. [15] and Trump lawyer, John Dowd [16], have authored tweets in Trump's name. These "imposter" tweets risk polluting the data set with non-Trump content.

Given the deficiencies associated with training on tweets or the president's book, we decided to exclusively limit ourselves to public speeches made by Mr. Trump since he announced his bid to run for president on June 16, 2015. Some of these public speeches had already been compiled in repository in [17]. Another repository of largely different speeches had been collected in a separate repository [18]. Unlike the first set of speeches which was a static collection in text format, the second set used a web scraper to pull tweets from the University of California, Santa Barbara's campaign speech archive [19]. However, the web scraper had significant bugs that was corrupting the speech output. We modified the script using Python's `BeautifulSoup4` package to properly extract the tweets. We then manually merged the two training sets verifying there were no duplicates.

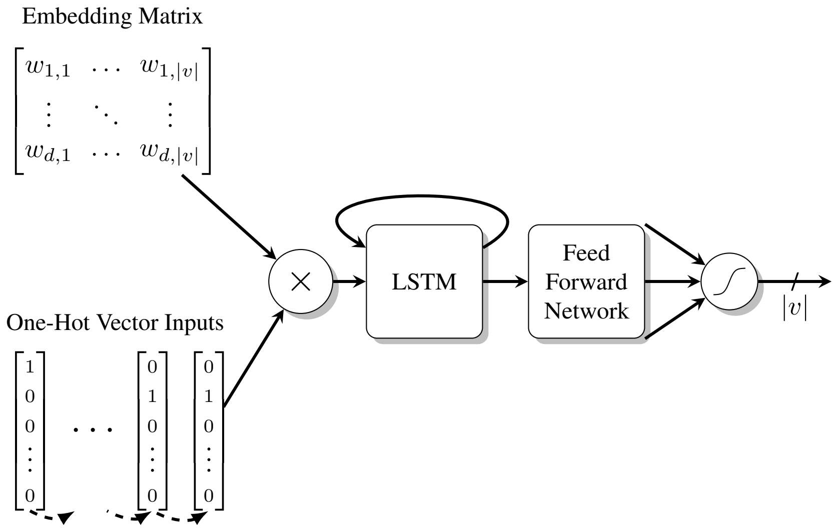
Embedding Matrix
$\begin{bmatrix} w_{1,1} & \cdots & w_{1,|v|} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \cdots & w_{d,|v|} \end{bmatrix}$
One-Hot Vector Inputs
LSTM
Feed Forward Network
$|v|$

Figure 1: Trump Character RNN Training Architecture

In total, the training set consisted of more than 115 speeches of varying length. There were more than 365,000 words and two million training sequences. We believe the training set size is more than sufficient for a project of this scope, and we are further confident this is shown in the quality of the generated output.

3.2 Vocabulary

As with all character-level RNNs, the vocabulary is set of individual characters that may be received as part of an input sequence or generated as part of the output. Specifically, the vocabulary consists of all letters – both capitalized and lowercase, digits (0-9), and punctuation (e.g., comma, space, newline, exclamation point, etc.).

For all practical purposes, the set of characters, v, that make up the vocabulary is dictated by the training data set. In this project, the size of the vocabulary, $|v|$,

3.3 Learner Architecture

3.3.1 One-Hot Vector Input

3.3.2 Embedding Matrix

3.3.3 Long Short-Term Memory

3.3.4 Feed-Forward Network

3.3.5 Softmax Layer

A softmax layer is function, σ, that a real-valued vector, $\mathbf{z}$, of fixed sized, $|v|$, and returns an equal-sized vector, $\mathbf{p}$ whose elements are between 0 and 1 inclusive. Hence, $\sigma : \mathbb{R}^{|v|} \rightarrow [0,1]^{|v|}$. The magnitude of each element $z_k \in \mathbf{z}$, is normalized to create value $p_k \in \mathbf{p}$ using the softmax function where:

$$p_k = \frac{\exp(z_k)}{\sum_{j=1}^{|v|} \exp(z_j)}. \tag{1}$$

Therefore, the softmax layer creates a probability vector as the sum of all its elements is necessarily 1. This standardized output is then used by the loss function as described in the next section.

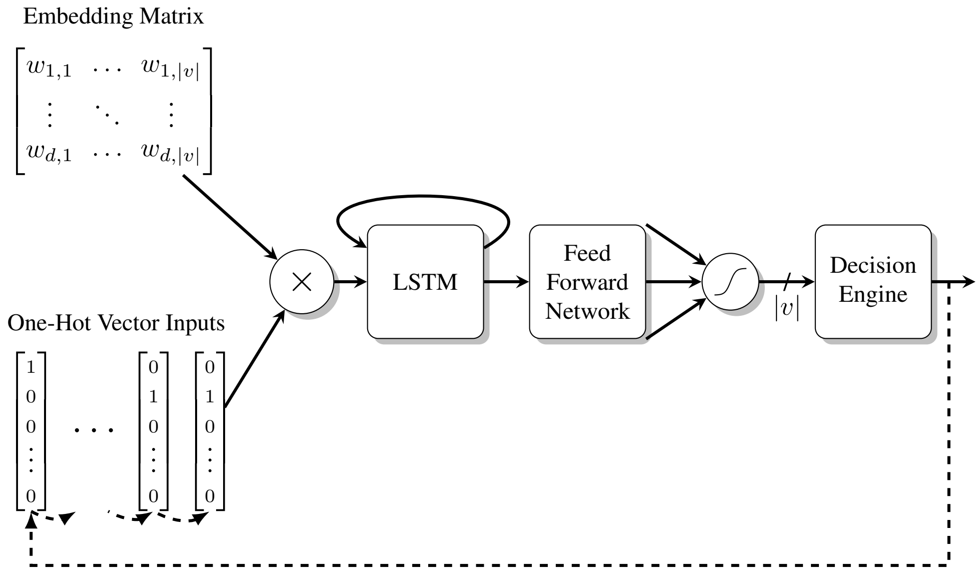
Embedding Matrix
$\begin{bmatrix} w_{1,1} & \cdots & w_{1,|v|} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \cdots & w_{d,|v|} \end{bmatrix}$
One-Hot Vector Inputs
LSTM
Feed Forward Network
$|v|$
Decision Engine

Figure 2: Trump Character RNN Text Generation Architecture

3.4 Loss Function

Our network trainer has two inputs, $X \in (V^L)^n$ and $y \in V^n$. X is an $n \times L$ matrix representation of our sequences, where n is the number of sequences and L is the number of characters in each sequence;] y is a vector of n target values, meaning that y_i is the character following sequence X_i. Let p be an $n \times |V|$ matrix where each row is a probability distribution over $V : p_i(y_i) = 1$.

Our loss per example is the cross-entropy between the distribution given by the network and the true distribution, which is one-hot. Formally, we say that loss on example $i = L_i = H(p_i, \mathcal{C}(X_i))$. Our total loss on inputs X and y is then $\sum_{i=1}^{n} L_i$.

4 Text Generation Architecture

5 Decision Engine

5.1 Greedy Sampling

5.2 Random First, Greedy Finish

5.3 Top-K First, Greedy Finish

5.4 Randomization through LSTM Dropout

6 Conclusions

6.1 Future Work

References

[1] D. Trump and T. Schwartz, *Trump: The Art of the Deal*. Random House, 1987.

[2] D. Trump and C. Leerhsen, *Trump: Surviving at the Top*. Random House, 1990.

[3] D. Trump and K. Bohner, *Trump: The Art of the Comeback*. Times Books, 1997.

[4] D. Trump and D. Shiflett, *The America We Deserve*. Renaissance Books, 2000.

[5] D. Trump and M. McIver, *Trump: How to Get Rich*. Random House, 2004.

[6] D. Trump, *Crippled America: How to Make America Great Again*. Threshold Editions, 2016.

[7] D. Trump, *Time to Get Tough: Making America Great Again!* Regnery Publishing, 2015.

[8] D. Trump and B. Zanker, *Think Big and Kick Ass: Make it Happen in Business and Life*. Morrow Avon, 2007.

[9] D. Trump and K. Bohner, *Trump: The Art of the Comeback*. Times Books, 1997.

[10] D. Trump and M. McIver, *Trump Never Give Up: How I Turned My Biggest Challenges Into Success*. John Wiley & Sons, 2008.

[11] D. Trump and R. T. Kiyosaki, *Why We Want You to be Rich: Two Men, One Message*. Plata Publishing, 2006.

[12] D. Trump, *Trump: The Best Golf Advice I Ever Received*. Crown Publishers, 2005.

[13] D. Trump and R. Kiyosaki, *Midas Touch: Why Some Entrepreneurs Get Rich, and Why Most Don't*. Plata Publishing, 2012.

[14] J. Mayer, "Donald Trump's ghostwriter tells all," *The New Yorker*, Jul 2016.

[15] A. Ohlheiser, "The (other) man behind the curtain of Trump's twitter account is revealed ... again," *The Washington Post*, Oct 2017.

[16] K. Phillips and A. Blake, "Trump on Michael Flynn's guilty plea: It's a 'shame' because he had 'nothing to hide'," *The Washington Post*, Dec 2017.

[17] R. McDermott, "1mb archive of donald trump speeches," 2017.

[18] P. Navid, "All of Trump's speeches from June 2015 to November 9, 2016," 2017.

[19] "2016 presidential election speeches and remarks," 2016.