Florida Man Returns: Generating New "Florida Man" Article Titles Using LSTM

Megan Hazlett
Master of Science in Analytics
Northwestern University
meganhazlett6@gmail.com
https://github.com/meganhazlett/FloridaMan

Abstract

By using LSTM and training on "Florida Man" articles scraped from Twitter and Reddit, we were able to generate new "Florida Man" articles based on user seed of 5 words. While the sentences generated by the model were quite incoherent, they did provide some interesting ideas for new potential articles. This project was limited by the amount of data we could collect (only 6,000 articles), so we would imagine that the results would significantly improve with a larger data set and more computation power.

1 Introduction

"Florida Man" is internet meme that describes the various news articles about Florida residents and ridiculous reasons for arrest. It gained popularity in 2013 and has since remained a popular topic of internet conversation. We believed that it would be humorous and entertaining to be able to generate new "Florida Man" article titles through the topics learned in the Natural Language Processing course. Since the content of these articles is usually nonsensical, the moderate success that text generation has seen in the past few years just may be enough to successfully generate passing "Florida Man" articles.

2 Related Work

Back in 2011, Ilya Sutskever, James Martens, and Geoffrey Hinton published a paper about the capabilities of using recurrent neural networks (RNNs) as tools for text generation. They demonstrated, that with the use of Hessian-free optimizers, they could achieve good success on the character level of text generation. In 2017, Pawade et. al had some success using word-level Long Short-Term Memory (LSTM) to generate new stories, by tuning parameters like batch size and sequence length. Their goal was to minimize train loss and achieve readability, to which they achieved a 63% accuracy using human evaluators. The most inspiring work for this project was done in 2015 by Peter Potash, Alexey Romanov, and Anna Rumshisky, who aimed to generate new rap lyrics using LSTMs. Due to the subjectivity of human evaluators, they decided to use cosine similarity to measure success. These researches used word-level data to not only generate believable lyrics, but also lyrics follow rhythmic and metric patterns.

3 Dataset

The data for this project was acquired through Reddit and Twitter's API's. By using the Reddit API, the researcher was able to scrape 2,811 "Florida Man" article titles from the Florida Man subreddit. Using the Twitter API, the research also scraped 3,204 article titles from the twitter user @FloridaMan__. This created a dataset of 6,105 articles. Although a larger dataset may have provided better results, this was the most amount of data the researcher could collect in the time frame given. Some examples of article titles include the following: *Florida Teen Arrested for a Chemistry Experiment Gone Wrong; Naked Florida Man Arrested for Punching Police K-9.* Code for acquiring the data can be found in the *get_florida_data.py* script on GitHub. Some article examples are as follows:

(1) “Florida Man ditches his family, starts a new life using a dead man's identity, busted when the dead man's nephew starts a family search on Ancestry.com”
(2) “Florida man busted in illegal, million-dollar flying squirrel trafficking ring”
(3) “Florida man dresses up as creepy clown to scare misbehaving children”

4 Method

Before even beginning the modeling process, we cleaned the data through removing hyperlinks and backslashes, converting all text to lower case, and tokenization. From there, a dictionary of words was generated and vectorized. This process involved the aid of packages and methods such as *nltk*, *numpy*, and list comprehension.

For the actual generation of text, the researcher ran an LSTM model with an Adam optimizer, softmax activation, and batch size of 128. We experimented with generating text on the character and word level, the sequence length (tried sequence lengths of 10 and 5), and dropout probability (20% and 60%). Ultimately, we found the best value in generating text from the word level with sequence length of 5 and dropout of 20%. The preference for sequence length 5 was driven by the user usability – users only need to provide a seed of 5 words to generate the rest of the article, which was set to be an additional 7 words. This model final was run on 1, 10, 50, and finally 100 epochs.

5 Results

After the final model was trained, we generated 100 new Florida man articles. While research suggested that the new article would be unreadable, with some cleaning, these articles show promise. The following are examples of articles generated by the model:

(1) with booze filled yeti cup florida man arrested for allegedly to segway library man
(2) from walmart naked florida man swims his drank to pirate smoke worldwide hillary mind controlled florida

With some cleaning, there is actually some merit to these articles. For example, the first article, ignoring the seed, the we can extract some interesting potential topics from the article. For example, the phrase “segway library man” could be written into an article like, “Florida man arrested for attacking library man on segway”. In the second article, we can extract interesting phrases such as “pirate smoke” and “Hillary mind controlled Florida”. Although the generated text is quite nonsensical, the nonsense provides just the right amount of randomness to drive some potential “Florida man” topics.

Of course, we need to acknowledge the notable pitfalls. First of all, there is an odd penchant for the word certain words generated in this result set. We observed that often, certain words appear over an over again. In some runs we have seen words like “romp”, “dough”, and “donor” come up repeatedly. In other result sets, words like “bread”, “face”, “manages”, and “gay” have appeared across most articles, and usually many times within the articles. To actually evaluate how good the model was, we compared two randomly selected real articles’ cosine similarity to the cosine similarity of one real and one generated article. The researcher then computed the average of 1000 of each. The cosine similarity statistic for two randomly compared real articles was 12.6%, while the statistic was 1.6% for the generated compared to the real article. This implies that our generated articles do not really resemble real articles.

As far as user generated work, the following are examples of seeds we tried and the results.

(1) “florida man arrested after he” -> swatted books florida man arrested for
(2) “parking lot florida man arrested” -> after purse to flee wal mart
(3) “jake miller walks to restaurant” -> carrying slowed man teases has to
(4) “aspiring local rapper finds florida” -> destroys knew man finger convention
(5) “category five hurricane florida man” -> wore after flinging to surveillance cameras
(6) “during hurricane 600 pound florida” -> man stealing reads roommate relax after
(7) “from pizza parlor florida man arrested” -> after sucker to ketchup drown packages
(8) “hi hi hi hi hi man” -> florida man hunting after florida

(9) “in a fit of rage” -> florida man arrested for allegedly to

(10) “doctor in florida region eats” -> florida man arrested for allegedly

6 Discussion

Overall, this project behaved in the way we expected with little success making coherent sentences. However, we were pleasantly surprised to find that some of the generated articles, while syntactically incoherent, provided some interested ideas for articles. Going forward, it would be very interesting to see if longer training and more data points could improve this. Since 6,000 data points in considered very small in term of deep learning project, we expect this to be so. Another interesting approach that could improve the results would be to try on bi-grams.

References

Brownlee, Jason. “Text Generation With LSTM Recurrent Neural Networks in Python with Keras.” Machine Learning Mastery, 2 Sept. 2020, machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/.

Pawade, D., Sakhapara, A., Jain, M., Jain, N., & Gada, K. (2018). Story scrambler-automatic text

generation using word level rnn-lstm. *International Journal of Information Technology and Computer Science (IJITCS)*, *10*(6), 44-53.

Potash, P., Romanov, A., & Rumshisky, A. (2015, September). Ghostwriter: Using an lstm for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1919-1924).

Sutskever, I., Martens, J., & Hinton, G. E. (2011, January). Generating text with recurrent neural networks. In *ICML*.

“Text Generation with an RNN : TensorFlow Core.” TensorFlow,14 Oct. 2020, www.tensorflow.org/tutorials/text/text_generation.