Generating Teamfight Tactics Tweets Using a Character-Level Recurrent Neural Network

Author

- Kyle Vu
- UC San Diego
- kdvu@ucsd.edu, kyledvu324@gmail.com

Abstract

This research addresses the unique vocabulary within the Teamfight Tactics (TFT) community of practice by training a character-level recurrent neural network (char RNN) to reproduce the specific language used in TFT-related texts. The primary data sources include tweets from popular TFT content creators and transcriptions of YouTube videos by the popular TFT player, k3soju. The findings reveal that the char RNN effectively learns and generates text that mimics the TFT lexicon when trained on a focused dataset of tweets, achieving a significant reduction in loss and producing coherent outputs. However, training on a combined dataset of tweets and YouTube transcripts resulted in poor performance, indicating the challenge of dealing with more complex and noisy data. This research contributes to understanding the complexities of the TFT lexicon and demonstrates the potential for char RNNs in generating unique output in specific fields.

---

1. Introduction

Teamfight Tactics (TFT) is an auto-battler game developed by Riot Games in which players build armies of champions which automatically fight against other players' teams over a series of rounds. It has a community of practice (CoP) with a notably unique vernacular consisting of gamer-specific lexicon not used in other gaming communities [1]. The vernacular of the TFT CoP allows them to communicate particular ideas and evoke emotions in ways not seen outside of this community. While there is some research regarding the unique language involved with TFT, there has yet to be any attempts to teach artificial intelligence to reproduce this language via character-level recurrent neural networks (char RNNs).

This work aims to train a character-level recurrent neural network on TFT-related text. Specifically, the data is sourced from Twitter accounts of popular TFT content creators (such as @k3soju, @MilkTFT, and @Rayditzfn) as well as transcribed k3soju's YouTube video captions. The primary goal is for this neural network to be able to properly learn unique phrases and vocabulary associated with TFT, outputting relevant text content that demonstrates an understanding of the phrases and words used within its community of practice. This can contribute to a deeper knowledge/understanding of the dynamics of the TFT lexicon as well as how to train/tune hyperparameters for char RNN models.

---

2. Method

2.1 Overview

In order to generate TFT text results, I trained a character-level recurrent neural network. Essentially, given a large amount of text, a char RNN is able to predict the next character in a sequence and thus create new text one character at a time [2]. The neural network will learn from input data, detecting patterns in the way words are spelled and phrases are grammatically formed. After training, we can continuously sample from the trained model to keep generating new output.

2.2 Data Collection

To collect tweet data of well-known content creators in the TFT community, I used the TwExportly Google Chrome extension [3] to scrape tweets and store them as .csv files. However, I was limited in the number of tweets I could collect primarily due to rate limitations. I initially attempted to use the Twitter/X API to collect Twitter data, but as of September 8, 2023, users on the Free developer plan are no longer able to use this API to read tweets [4].

In addition, I collected YouTube transcript data to supplement the tweet data and provide the model with more natural, spoken TFT language usage by the most popular TFT content creator k3soju. The raw, auto-generated transcripts were not suitable though since they were not able to recognize TFT-specific language (mistranslating phrases like "greeding" → "greeting", "eif" → "eth" etc). To circumvent this, I began by initially doing a manual correction process of 10 of the 42 most recently uploaded videos on this channel (correlating with videos that depict the most recent version of TFT gameplay). I compiled a dictionary of common mistranslations across videos and used this dictionary to correct the remainder of the YouTube transcripts. Besides this, I did some simple data processing such as removing meaningless text in the transcripts ("[Music]", "[___]", etc).

2.3 Model Architecture

The model architecture of my character-level recurrent neural network is based on Nikhil Barhate's PyTorch implementation [5]. It is a minimalist code structure using Multi-layer Recurrent Neural Networks with Long Short-Term Memory (LSTM). The initial text input is first passed into an embedding layer, converting each character into a dense vector of a fixed size. Then, the data passes through a number of LSTM layers (the number of which depends on hyperparameter selection, but typically 2 or 3). These layers are aimed to help alleviate the vanishing gradient problem to ensure the RNN captures long-term dependencies which are important for language processing tasks. Lastly, the LSTM output passes through a linear decoder layer which produces logits for each character in the vocabulary.

The training process involved feeding sequences of characters from the TFT text data into the aforementioned stacked LSTM architecture. In terms of hyperparameter selection, each input sequence was of length 100 characters and each of the three LSTM layers contained a hidden size of 512. The model is trained to minimize Cross-Entropy Loss using the Adam optimizer with a learning rate of 0.002. During training, sequences are sampled from random starting points in the text to ensure learning of the entire dataset occurs. After each epoch, the model's weights are saved, and a sample text sequence is generated to allow for continuous assessment of the model's performance. This training continues for 100 epochs to allow the model to learn and generalize the character distributions and structure of the text.

---

3. Experiment

3.1 Experimental Setup

I experimented heavily with the model architecture, hyperparameters, and dataset throughout my process. Between the combined Twitter data as well as the YouTube transcript data, I had around 800,000 characters. However, to create a more balanced dataset and weigh my model more heavily towards creating text in the Twitter format, I duplicated my Twitter text 3 times to train on a new dataset closer to 1.4 million characters long. It is important to note my most successful results came from using a smaller, more focused dataset and staying close to the default hyperparameters in Nikhil Barhate's implementation.

To make more usage of my computer's GPU, I implemented batching which allows for multiple text sequences to be processed simultaneously, allowing for significantly faster model training. I also attempted to use a scheduler to adjust the learning rate by decreasing it when the model encountered plateaus in its learning.

I attempted to train my model with different optimizers (like Stochastic Gradient Descent) and adjusting the number of layers (2 instead of 3). I also experimented with changes to the learning rate (from 0.001-0.004), different sizes of hidden layers (128 and 512) and different

sequence lengths (50, 100) for the hyperparameters. This process of hyperparameter adjustment was somewhat arbitrary, but I attempted to follow along with recommended practices based on Sherjil Ozair's char-rnn-tensorflow implementation [6]. Lastly, I adjusted the size of my dataset (number of characters total, number of unique characters) and the type of data within my dataset while I was training the model. I experimented with a smaller dataset containing just text scraped from Twitter accounts as well as the full dataset containing a mix of tweets and YouTube transcript text.

This experiment does not include training and test splits, as the goal was not to attempt to have the model generate text similar to a validation text. Rather, the goal was simply to qualitatively observe trained model outputs and determine if they are able to generate output that demonstrates an understanding of the unique TFT lexicon.

3.2 Results and Discussion

Firstly, it is important to note my most successful results came from using a smaller, more focused dataset and staying close to the default hyperparameters in Nikhil Barhate's implementation. This means a learning rate around 0.002, a hidden size of 512, 3 layers, and a sequence length around 100. Additionally, the results I interpreted to be the best came from training on the pure tweets portion of the data, which is a smaller ~300k characters dataset compared to the larger combined dataset. After training for 100 epochs, the loss was reduced from 3.59 down to around 0.289. Here is a small 1000 character sample from my trained model after 100 epochs:

```
VyKPwi"
@ineekoh @Uself @DeliciousMilkGG @k3soju

IN OTHER NEWS, LETS GOOO SOJU WOOOO https://t.co/dFqbQzynowown with @Kiyoon  wollom ASSIKE YOU MORT. TFT STILL THE BEST GAME CREATED AND SET 9
d"
@k3soju @Purilly @Prestivent @Prestivent you got the vegas to take him hit that serious, and every single bug gets fixed in everyone in your t
these bots are tucked"
@WARDELL416 Hurry Me sucfund If its something stacked with 15 minutes to enter a little legend giveaway and can't him deathwing because he bri
@MilkTFT How about y00 they did traitey. He can be your instagram bf

Cons:
Can't get drunk, but will always be down to grab a drink with you                Hating Yuumi"
@MilkTFT 2 star jeffrey pan turned into 3 star DeliciousMilk
"trying to think the game by itself is also in itself a content creatio
```

Admittedly, the novel value in this generated text is dependent on one's knowledge of the TFT community and culture. Even without this knowledge, though, it is clear to see that my model was able to learn the style of tweets and the English language in general, creating fake links in the style of real links and using the @'s of real handles in the TFT Twitter community.

My attempts at training on the more complex dataset did not yield valuable results. My model was essentially unable to learn at all, converging with a loss value above 3 within 5-10 epochs consistently even with hyperparameter tuning. I believe this was simply due to the much higher complexity of the data when using a combination of Twitter and Youtube transcript data. By introducing the transcript data which essentially involves a person speaking freely, often

about various topics, the data was likely too noisy/messy for the model to be able to learn any concrete patterns. Essentially, my results at training on this dataset only yielded gibberish that did not resemble TFT language nor even English.

---

4. Conclusion

4.1 Summary

My research somewhat successfully trained a character-level recurrent neural network (char RNN) to generate text reflective of the unique vernacular used within the Teamfight Tactics (TFT) community. By leveraging data from Twitter accounts of well-known TFT content creators and transcribed YouTube videos, the model learned to produce text that closely mimics the language used in this specific community of practice. The most successful results were achieved with a smaller, focused dataset of specifically tweets, with the model demonstrating a strong grasp of the TFT lexicon and significantly reducing the loss over 100 epochs. However, attempts to train the model on a combined dataset of tweets and YouTube transcripts were less effective due to the increased complexity and noisiness of the data.

4.2 Future Work

Future research could do several things to improve upon these results. One potential direction that could be taken is to refine the data preprocessing to better handle the mixed dataset, potentially improving the model's ability to learn from more various sources. Additionally, experimenting with more advanced neural network architectures might yield better results depending on their ability to learn from text data such as the data in this research. Another area for exploration is fine-tuning the hyperparameters and incorporating techniques such as data augmentation to further enhance the model's learning capabilities. Finally, expanding the dataset to include more varied sources of TFT-related text could provide a better background for training, enabling the model to generate even more accurate and diverse outputs reflective of the TFT community's vernacular.

References

[1] Aa, J. T. A. van der. (2021, July 1). *"it's a First!": A case study on identity expression through language on Twitch in a game of teamfight tactics*. Utrecht University Student Theses Repository Home. https://studenttheses.uu.nl/handle/20.500.12932/40394

[2] The unreasonable effectiveness of recurrent neural networks. (n.d.). http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[3] Google. (n.d.). *Twexportly: Export tweets from any account*. Google. https://chromewebstore.google.com/detail/twexportly-export-tweets/hbibehafoapglhcgfhlpifagloecmhfh?hl=en

[4] *Changelog*. Use Cases, Tutorials, & Documentation. (n.d.). https://developer.x.com/en/updates/changelog

[5] nikhilbarhate99. (n.d.). *Nikhilbarhate99/Char-RNN-PyTorch: Minimal implementation of Multi-layer recurrent neural networks (LSTM) for character-level language modelling in pytorch*. GitHub. https://github.com/nikhilbarhate99/Char-RNN-PyTorch

[6] Sherjilozair. (n.d.). *Sherjilozair/Char-RNN-tensorflow: Multi-layer recurrent neural networks (LSTM, RNN) for character-level language models in python using tensorflow*. GitHub. https://github.com/sherjilozair/char-rnn-tensorflow

---

Supplemental Materials

The code and data used for this research are available on GitHub: https://github.com/kyledvu. This repository includes the data collection scripts, the implementation of the character-level recurrent neural network, and the trained models.