CS 6120: Natural Language Processing
Spring ’22 - Project Report

Text Generation Using RNN

Authors

Pankaj Pandey, Yash Bhojwani, Uma Patil, Yashvi Bhandari

Abstract

Text generation is a task in Natural Language Processing where text is generated with some constraints such as initial characters or initial words. A deep learning model is trained to generate random but meaningful text in the simplest form. It can be seen in our day-to-day applications such as character / word predictions while typing texts in Google browser, Gmail, Smart Keyboards, Multi-document Summarization / Compression, etc.

Introduction

Text generation is a very important feature for AL-based tools. It is very useful in machines that are supposed to become more interactive with humans such as smart gadgets. Therefore, in this project, a new script for the Friends TV series is being created from the “Friends Transcripts” dataset using Recurrent Neural Networks (RNN). A recurrent neural network is a type of artificial neural network which uses sequential data or time-series data.

Dataset Description

The dataset used contains the script for all the seasons of the famous TV series friends. The link for the dataset is - https://fangj.github.io/friends/. It has 10 seasons, and each season has almost 24 episodes. Each episode redirects to the script which contains the basic information about the episode, the title of the episode, the character names, and their dialogues in order. The dataset was presented in an unorganized way and hence to make use of it, the dataset was structured and formatted properly.

EDA and Data Preprocessing

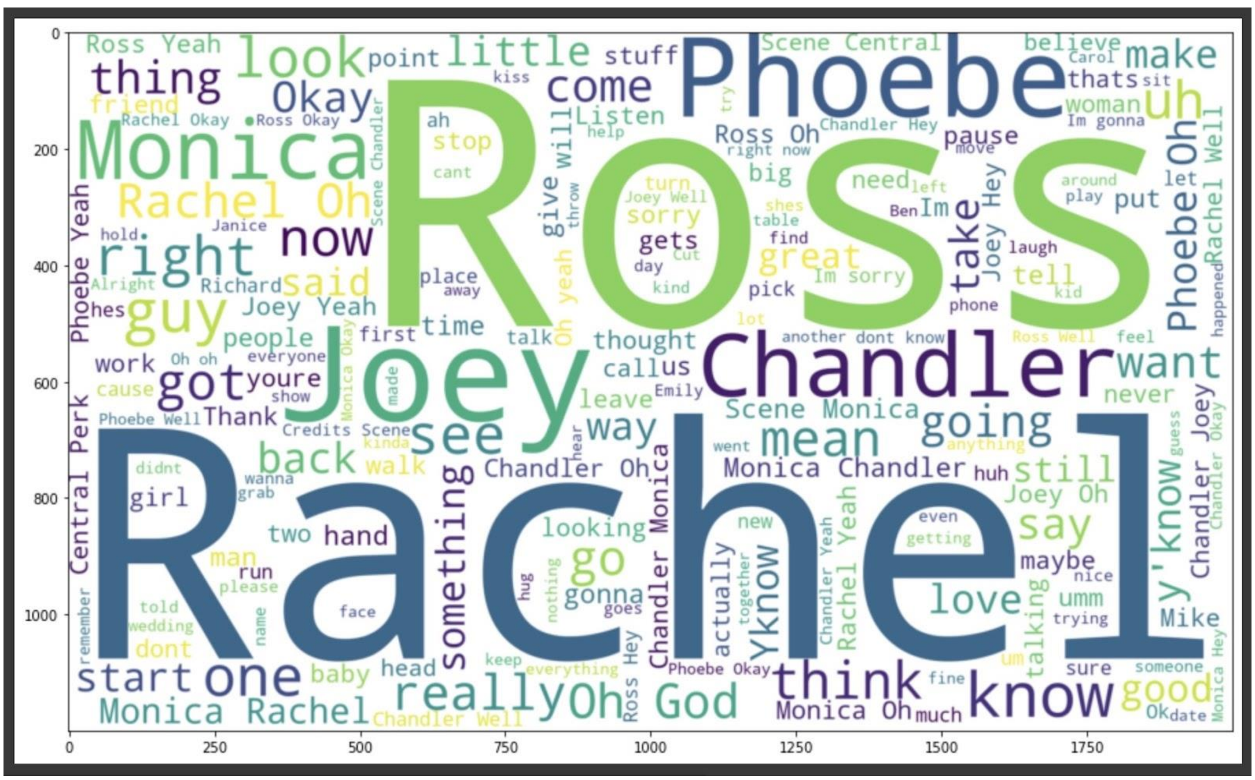
Firstly, the metadata about the scene from the dataset was identified and the length of text was calculated, that is the number of characters in it. A vocabulary to get the unique texts in the whole dataset was also created. Later, the most frequent characters/words in the dataset were calculated. To fit the data in its respective model, data in the form of dialogue conversation with the name of the character separated from the “:” we needed. Hence all the other metadata about the scene from the dataset was removed. The sentences were tokenized, and index values were assigned to the characters. Removing unnecessary words and appending appropriate sentence start and end tags were the next

steps. An embedded weight matrix for the vector representation of the data was also Created to get some distinct features out of the text for the model that was being trained.

It started with the creation of a Data Loader Class. The path variable was set, and the data variable which consists of the input transcript data, and vocabulary which is the list of all the unique characters in the dataset were created. During the Data Exploration, it was found that there were 94 unique characters to be precise including all the alphabets, numbers, and special characters.

Next, the characters to integers were mapped to create the vector that was fed to the model during training. Also, a dictionary that stores the integer to character mapping was created so that while receiving the predictions from the model the vectors could be converted back to their respective characters. Additionally, since an RNN was used, data was to be fed of similar lengths therefore the data was formatted in such a manner that the data was fed in the length of 100 characters. Later the input and target data used while training the model were also initialized.

In the exploratory process, a word cloud to visualize all the frequent words in the dataset was created. The image given below is the word cloud that was created.



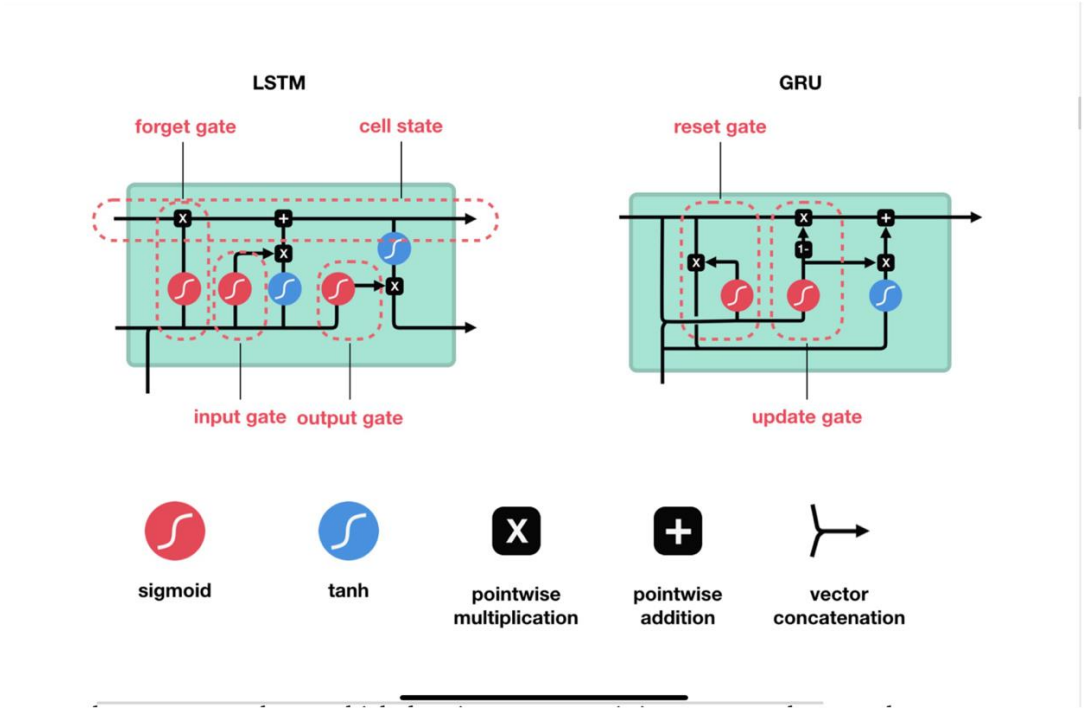
Modeling

The model consists of a RNN with an embedding layer , a GRUnit(running with 256 RNN units) and 3 sense layers that feeds the data in predetermined batches of fixed

size. The model is trained by varying hyperparameters, batch sizes, embedded weights, and model layers to obtain the highest performance. Finally, a final graph is utilized to create a word picker function that takes a word and finds the probability of the following word, provided a word that serves as the sequence's start.

Summary of the model:

```
Model: "script_generate_model_28"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_41 (Embedding)     multiple                  12032

 gru_40 (GRU)                 multiple                  296448

 dense_56 (Dense)             multiple                  24158

 dense_57 (Dense)             multiple                  8930

 dense_58 (Dense)             multiple                  8930

=================================================================
Total params: 350,498
Trainable params: 350,498
Non-trainable params: 0
_________________________________________________________________
```



Results

Consists of a brand-new script that is generated by training the LSTM model. The script is generated for an episode with meaningful dialogues and character representation.

Future work

The future work for this project would consist of using one of the famous deep learning models known as Generative Adversarial Networks (GANs) as it is very different from the conventional models, as they possess an adversarial way of training the network. We will also try using another famous model called Variational Auto-Encoder (VAE).

Conclusion

Deep learning provides a way to strap huge amounts of data with ease. Therefore, the help of word embedding and popular models like CNNs, RNNs, VAEs, and GANs have made the Natural language processing problems a bit easier. However, the current deep learning models have not completely captured the technicalities and interpretation of natural language and require a better metric generated by humans for the evaluation progress of the text generation. This project uses only RNNs for the generation of the new script and requires using the of more advanced models like GANs and VAEs further.

Statement of contribution (Works cited)

There are four members of our team: Yash Bhojwani, Yashvi Bhandari, Pankaj Pandey, and Uma Nishikant Patil. We have equally distributed the work amongst each other.

Yashvi Bhandari was responsible for cleaning, preprocessing, and performing the EDA of the data. Yash Bhojwani and Pankaj Pandey trained the RNN model and compared their performances. Uma Nishikant Patil improved the model by tuning hyperparameters and by making adjustments to model weights and layers as well. The documentation was prepared together and discussed each other's part as well.

References (Supplementary material used for project)

- Loannis Konstas and Mirella Lapata. 2013. A global model for a concept-to-text generation. Journal of Artificial Intelligence Research (JAIR), 48:305–346
- Explaining Text Generation with LSTM – Analytics Vidhya
- Generating Text with Recurrent Neural Networks [pdf], 2011

Github Project Link

https://github.com/yashvii5/Text-Generation-Using-RNN