Text Generation Using
Long Short Term Memory Neural Networks

1. Introduction

As learned in the previous writeup, LSTM's have significant ability to predict sequential data. In this experiment, poetry and rhyme data is used to create a model that can generate its own poems.

2. Data preparation

2.1. Data Cleaning

The data preparation is where we format the data to be in a machine learnable format. The data is initially in a tabular form where the first column is the title of the poem and the second column contains the text in the poem. The data is originally scraped from a tab formatted text file. This text file also had different lines and verses, but because this model is a preliminary model, I decided to exclude this information and build a single string containing all the lines and verses together. I also cut out all special characters. Though punctuation would be interesting, I decided to keep normal lowercase letters and spaces. This would require separating punctuation from words, or else "cool" and "cool." would be seen as different words.

2.2. Tokenizing

The next step is to tokenize the data. At this step we have all the poems combined into a single string. Letters and words are not intrinsically usable by a machine. At this step we tokenize each individual word. Splitting the string data by spaces allows us to separate all the words. Then on those words we tokenize. What is tokenizing? Tokenizing takes the dictionary of words we have and assign a number to it. "Cool" might be assigned with 1 and "beans" might be assigned with 5. The number goes up to the number of unique words in the string. After this the tokens are able to be passed into a model.

2.3. Sequencing

The Data is sequenced by running through the encoded words with a for loop and taking a range of 3 to be put into the data. The model needs equal length inputs, so we will pad all the words to the same length as the longest one. After this the sequences will be split

into X and y data. X is the input and y is the output. The y output is then split into a categorical variable. This way the model can use categorical predictors to predict instead of predicting on the arbitrary value associated with it in the tokenizing.

3. Modelling

3.1. Overview

The modelling approach is inspired by a machine learning mastery article attached in the README (in the GitHub). The model uses n inputs to predict the next word in the sequence. This model uses the last two words as input and predicts the next word. Each line of this model uses two random words to do the generation.

3.2. Neural Network Geometry

The network is sequential. The three layers in it are an Embedding layer, a LSTM layer, and a final dense layer to build a probability distribution across all of the words in the input data. The Embedding layer is a way to build what the most and least important words in that data are. Word2Vec is something that is used here, but in this model, we build it ourselves. It is about the ‘relatedness’ of words. The next layer is an LSTM layer. This layer has 50 LSTM nodes of input that learn the internal weights from the data. The final layer is the dense layer. This is the size of the vocabulary. It returns a probability distribution of all the words. In prediction we only take the most probable word, but we can also take a random choice from the provided probability distribution.

*Aside (I am not sure if the 50 units in the LSTM layer represent 50 LSTMs of depth 2 in parallel or represent 50 single LSTM units)

4.Generation

The generation of the poems is relatively simple. The model is provided with two words as a seed. You can put in the words yourself or in this case, select two random words from the data. If you input them yourself, the words inputted might not be in the dictionary that the model trained on, so it is likely that the model doesn’t know what you are talking about. It will still build a poem without those as well.

Here is one example of a poem generated by the model:

called she pin all and he one fly bleed joyful wedding when neer going go dine mother says a on if
to children as he her hop i and tune milk said night and your then and white at come think love
a rigs going says a ate nothing often and will may write made barley sake all to hop you up buble
her hay sun then frog or the and the upon to swim woman notshe well go will the all burned take
say clout hole either as john as her all thou by the the and lift any and little young went as
it both song up full will a man them miller spoke three or his together do rat and you made it
way put the i little thou wont quoth joy what man and matter in was at he do quiet they and
milk bridge and i a a up the the he a mother she broom his remedy have the and silver safe
a the clothes his school is that hill his it he he how ride off came wand lady to and could
upon said i the had made may clean little black the lived and yellow for down them dogs his is gave
the down side little cant nail little gone little i every are nest that the to look want out yonder the
applepie has with you your he pulled mother cats did i one keep dig dolly if news there said and for
was she carrion walls locket a cock behind but got or little rope merry for jolly she a iron her the
drums frog swarm purrr i on the it not there it the and suns plum the what in such new drink
shall an that he carrion must bread a be ho cake bit queen what the got his soon penny if gently
wont were came over on dame he would it raised the it thief their purrr mammy ah found and the wheelbarrow
water x a a old the she one the she pig jenny too eight the he pig his on care not
he betty the to going pray you burden nanny high dont little to i maids the a of with why clothes
dressed bit away and loved do dont try fell my good your in raven pair through three way mice was very
sea get and b a old took oh of i out tune a head miller down me run fire reason fire
and my a quoth the remain and burden calf mother he burden for both little a of make for i the
to to so taffy she it to loved her will not wilt see says nancy was of if sung money up
was this heigho the but kill with jack miller both he brought cats going and i here about mother all find
the with and to tails cow not in out creep go with was heigh pigs heigh gave the had and killed
the made his came little too do my and will robin the but her then is the found the it again
flew neither there moon or to i the little and and the toes and over little he was you you battle
back some do it heigh a and a minds h to clothes plainlooking to nobody heigh i kittens going old you
wives mistress you i fiddle sing robin tail and was the and the pig carried is man and a pig a
mittens there pussy and she kittens tried pounds not ill the is down cook eho can ding with old storms for
making whip it does carried and to about and penny a this out at black news two of him with you

Does it makes sense? No, but it is interesting. There is no punctuation or grammatical checking, but it is able to *kind of* write a poem, maybe just a slam poem. This model only uses the two previous words, it doesn't use context or punctuation or grammar, so the output is rough. I notice after generating the models that the words are spelled, *interestingly.* Either a typo or intentional the model doesn't see the difference. Along with the model being *'*dumb*',* it has a possibility to repeat after poems that exist. Because it is using the last two words, if they match up properly and the model is overtrained on those two words, it can end up generating a sequence that matches the poems that it took in while training.

4.1. Random Generation

This model selects the most probable word to write. This can be changed to select the word based off of a random choice in a distribution. If there was a bag filled with 3 red balls and 1 black ball, the model will pick the red ball 100% of the time. If the model was

allowed to pick from a distribution, it would pick the red ball 75% of the time and the black one 25% of the time given enough picks. This introduces some randomness and what I think is a human aspect of the model. This way, given the same input, it won't write the same thing every time. I didn't do this, but it should be attempted to see how the output changes. Given correct grammar and punctuation it should be even more obvious when this occurs.

5. Conclusion

Generative LSTM models are fascinating. As much as I love linear models, there are some data types that struggle to work with them. The multi layered approach in this model allows it to learn certain patterns and generate some interesting and odd texts. Just in this project there are a lot of things that can be changed to make the output more realistic and humanlike. Some of the few things I would like to change are noted below. Another limiting factor of this model is that it cannot extrapolate beyond its input. It has a range of what it can generate and decipher and it can't do anything beyond that. Machines are only as smart as their creators.

5.1. Future tasks

5.1.1. Build this model into a class structure for future usability. Build The differentiations in this into the class and functions.

5.1.2. Build off of verses and lines to build a more in depth generative model.

5.1.3. Build random word selection into the generation method

5.1.4. Use punctuation in prediction