LSTM Modeling (Assignment 4)

Dexter Corley

Argyros School of Business & Economics

&

Fowler School of Engineering

CPSC 393 - 01

Dr. Nicholas LaHaye

April 20, 2023

Abstract

This assignment writeup covers two separate LSTM tutorials along with our learnings from them. Additionally, the second part outlines creating our own LSTM models to be trained using nursery rhymes with the end goal of making our own.

Introduction

A LSTM model is great at taking time series data and returning an unknown future series of data. LSTM stands for long short-term memory which describes the ability for this algorithm to have a memory for what is deemed important. In the case of this assignment, we focused on three different things, multiclass text classification, regression time series, and generative text. The multiclass text classification model was built from a tutorial on medium outlining how the model works, and an example use case through reading articles and classifying the genre. The regression time series models were comprised of multiple different ways to use a LSTM time series regression with different nuances to help the model. Lastly, the nursery rhyme generative model was trained through a text file of hundreds of nursery rhymes and fed a starting sentence. All were different examples of ways to use LSTM models.

Body

Multiclass Text Classification

The multiclass text classification model has the main goal to correctly categorize articles that it reads. The training is done on labeled articles and undergoes various preprocessing methods that enable to the model to understand the data. The data is read in then goes through a series of processes including removing stop words like (in, is, to, the) to speed up the recognition. Additionally, tokenizing the sequences of words to their respective number then turning these sequences of numbers and words just to sequences of numbers. Then building the model and fitting it to the data. The model is built with dropout and a dense layer with bidirectional movement, so it gets trained front to back ensuring the long range memory is correct from the

beginning to end and vise versa. Once the model is fit and trained, we build out the code to predict the genres. Once ready, the user feeds in text from an unseen article and the model can correctly stamp the genre most of the time.

LSTM Regression Examples

The LSTM regression model tutorial was comprised of five different variations of the model. It began with an out of the box basic time series regression, then one using the window method, next using time steps, and finally stacked and unstacked memory between batches. The goal of each model was to forecast the number of passengers who would buy tickets the next month. Each of these are similar yet have some differences that work better or worse given different situations. To begin, the standard regression time series model takes historical data and predicts the projected amount of future ticket sales. The resulting error from the test set of this model came out to be 49k passengers. This isn't great but its also not terrible given the scale of the data. Next, the window takes in recent timestamps within the set window size where the model uses more recent data within the dataset to derive its predictions. Though the error was significantly worse at about 70k, it wasn't properly tuned and could lead to better results when more time is spent tuning the model. Next, using time steps can be another way to predict future results based on more recent data. Time steps are similar to the window method in the fact that it uses more recent data. However, instead of taking a broad window time steps takes the time – 1 to predict the future result close in time. Basically, using the most frequent data to produce the very next output. This resulted in better results than the window method at 60k RMSE, however is still slightly worse than the normal time series regression. Lastly, controlling

the memory between batches is another way to fit a better model to the data. This is done by changing the LSTM layer to be stateful and requires the training data to not be shuffled. Resulting in better long-term memory over the whole model and a RMSE score of 49.55k which rivals the normal model. In addition, you can also stack this approach into a deep network. This gives the model more depth and should be more accurate with more epochs of training. With only 100, this model garnered a 55k RMSE which again, is not great but getting close.

Generative LSTM Nursery Rhyme

The final part of this assignment was to train a generative LSTM model from a text file containing hundreds of nursery rhymes to create a nursery rhyme of its own given an input sentence to branch off. This assignment proved to be quite difficult and still didn't garner a great result. To begin, the text file consisted of nursery rhymes that were laid out as follows: the title, 2 blank lines, the nursery rhyme body, 4 blank lines, then the next one. All repeating that same pattern. The exception for this is that some of them didn't contain just words and had boxes or drawings in them. To account for this, the loop skipped over these when encoding. For the model to read each one, each rhyme had to be separated into its own dictionary with the title being the key. After this was sorted, it needed to be tokenized so that the words were numbers, allowing the model to be able to draw patterns between the sequences. Once the sequence length parameter was calculated, the model could be built and fit to the data. After fitting, in order to generate a new nursery rhyme, the generate rhyme function was created. This takes in the seed text as well as defined number of lines and words per line to use in the output. Finally resulting in a text file containing the output nursery rhyme.

The given input seed was: “Harry and Greg drank from a keg”

The output nursery rhyme was: “Harry and Greg drank from a keg hog it in buy do the let it in by fine dog maids and lives is wang on too jill caper you lady and them hat your lol them they.”

The results will be further examined in the conclusion section.

Conclusion

The generated nursery rhyme from this model is a long way away from being considered an English sentence, and lightyears away from being considered a nursery rhyme. To conclude, there are a couple of things I think made this model fall short. The first being how many epochs were trained with and the second being that I couldn’t get the output to be exactly 30 lines comprised of 20 words each. In regard to the former, I think if the model trained longer with more epochs there would be a better result. I will put this on the server with a lot more and see if the result is better. This can’t be done now though as I am getting an error saying the server doesn’t have numpy. Will submit a revision if the results are better in a couple of days. Lastly, I wasn’t able to get the model to output a nursery rhyme with the correct dimensions. I tried to update the code as best as I could but wasn’t able to get the desired result. Will also keep working on it and include any revisions in the next submission in a couple of days.

Citations

Brownlee, J. (2017, September 28). Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras. Machine Learning Mastery. https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

Jajafer, D. (2019, December 3). Multi-class Text Classification with Keras and LSTM. Medium. https://djajafer.medium.com/multi-class-text-classification-with-keras-and-lstm-4c5525bef592