Assignment 4 Write-Up

Introduction

LSTMs (Long Short Term Memory) are a special kind of recurrent neural network. Recurrent neural networks are very effective when processing sequential data. They take the output at a certain time and feed it back into the architecture. This way, we can have a look back at the model's previous behavior. We can take into account sequential and time based and relationships unlike other types of neural networks. LSTMs provide an additional benefit in that they utilize gates. These gates determine how much information should be forgotten or remembered and passed onto the next wave of training. LSTMs are often used to accomplish problems dealing with things like forecasting, text classification, and sequence generation.

Part I – BBC News Classification

Dataset

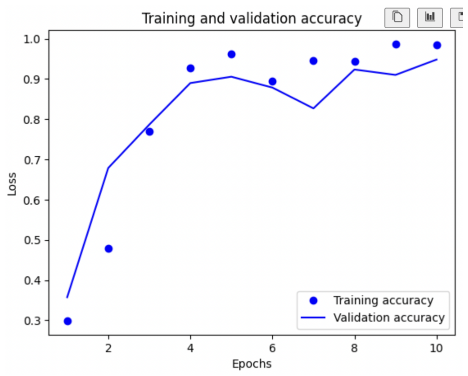
The BBC News Articles dataset contains a variety of articles that can be classified as one of five types of categories: business, entertainment, politics, sports, and tech. In order to get the data ready for the model, the "stopwords" were removed from the articles. Stopwords are a list of conjunctions and other words in the English language (ex. "In", "the", "because") that don't add value or are important to the model. We only want to work with key work with key words. After removing the stopwords we have to tokenize the remaining entries. Tokeneizing assigns a numerical value for each of the words remaining in the vocabulary. If a word is still not within the vocab, they are assigned the oov_token. The text is then transformed into sequences with their respective tokens and are padded so that each line is the same length as the others.

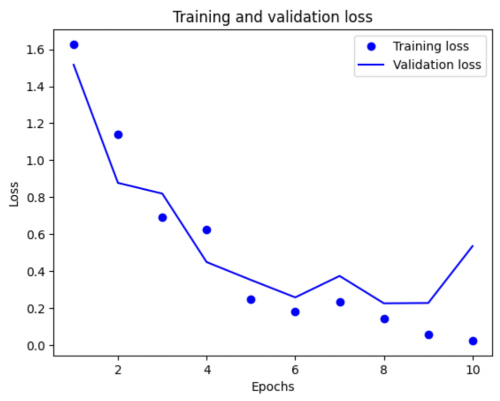
Method

The data was split into train and test data, with training containing 1780 observations and validation with 445. For the model, a sequential model was created. Like mentioned in the introduction, a sequential model helps to account for data that is influenced by the observations that come before it. The model consists of an embedding layer which represents a word as a vector. It also includes a dropout layer, set to 0.5, so that only half of the nodes are activated, helping to decrease overfitting. A bidirectional layer also runs the inputs in both the forward and backward direction, helping to generalize the model. Finally there is a dense layer with a softmax activation to predict the category of the article. The model was then compiled using categorical cross entropy as the loss function, adam for the optimizer, and accuracy to track performance. It was trained over 10 epochs and the model was then used to predict on the test data.

Results

While we can see that the model isn't 100% accurate, it is doing pretty well. On the training data it has about 99% accuracy and the validation data shows 90%, in that it is able to predict the correct outcome the majority of the time. The two graphs below also show that the model is likely not overfit and the model is pretty generalized, with both the validation and the training data trending in the same, positive direction.





Part II – Airline Passengers Prediction

Dataset

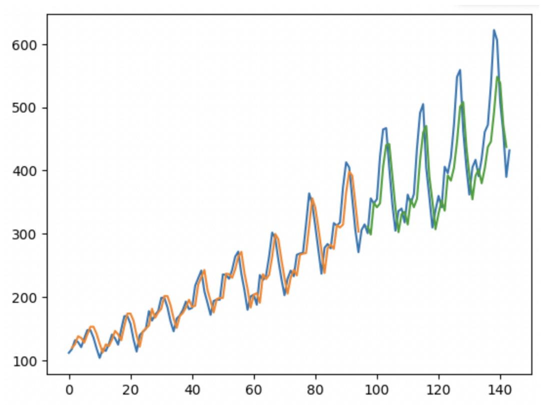
The International Airline Passengers dataset contains information in two columns; one with the month and year and the other with the number of passengers. To preprocess the data, MinMaxScaler was used to normalize. Additionally, the array of values were converted to a dataset matrix. To accomplish this, a function is created. The function takes a dataset as input along with a parameter look_back, which determines the number of previous time steps to use as input features (dataX) to predict the next time step (dataY). It was also reshaped so that the input follows the shape [samples, time steps, features].

Methods

For the model, a sequential model was used. The model has an LSTM layer with 4 memory units and it takes an input sequence with the dimensions of (1, look_back), where look_back = 1. It also has an output of 1, making one prediction given an input. It uses mean squared error as the loss and adam for the optimizer. There are also 100 epochs. The data was trained on the model and then used to predict the number of passengers using the train and test data.

Results

The model tells us that there is a RMSE (root mean squared error) of 22.64 on the training data and 49.87 for the test set. This is pretty good considering the range of values. Additionally, the graph below shows that the model is doing a great job of understanding and picking up the patterns of the data. It is a fairly generalized model and is able to accurately predict the number of passengers. While the model doesn't have as high of predicting power on the test data, the shape of the output follows the same pattern for both the training and the test data, implying that it is a successful model overall.



Part III – Nursery Rhymes Generation

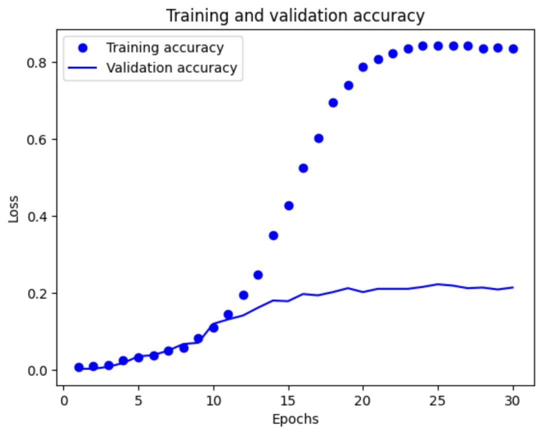
Dataset

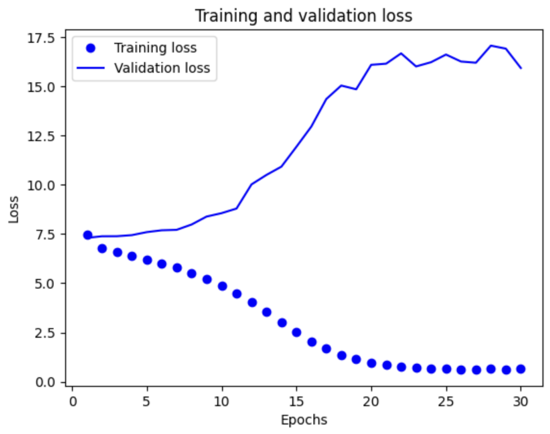
Nursery Rhymes dataset contains a variety of nursery rhymes. Initially, these rhymes begin with a title, then a double "\n", and followed by the rest of the rhyme. A new rhyme begins after a triple "\n". In order to clean and preprocess the data, I first began by using the .strip() function which removes the leading and trailing whitespaces. Alongside that, all of the letters were changed to lowercase to further maintain consistency. A remove_punc method was created to remove unwanted characters from the poems, specifically punctuation. Each string of words also became its own row in the list, so each line was separated instead of one long list. From there, all of the STOPWORDS – words like "and", "or", "in", etc. – were removed. These are not as important to the model and we want to focus on the key words. From there, each word was tokenized. Tokenizing means assigning a single number to each unique word. Finally, the rhymes were padded so that each line has the same number of characters in each. For the lines that have fewer than 20 words, additional zeros were added to the front so that equal length sequences could be fed through the model. Padding can be done either “post” or “pre”. I choose “pre” so that the lookback window in the model should mostly be viewing tokenized words and not padded zeros. The sequences were then split into X and y sets. The X set contains all words in the rhymes minus the very last column. The y set only contains the very last column. The X and the y were then sent through a 80/20 train_test_split.

Methods

To build the model, I used a sequential model with an embedding layer, a LSTM layer, and a dense layers. The embedding layer takes input dimensions equal to the size of the vocabulary (all of the potential words that could be outputted) which are 2322. The input length is 19 as the input is the size of each padded line, minus the very last word which the model aims to predict . For the LSTM layer, it has 128 nodes and an input share of (1, look_back). The look_back window is equal to 10 so it views the previous 10 words. I chose this number because while the input shape is 20, most of the lines have many leading zeros and the true sequences are only 5-10 words long. I also added a dense layer which uses relu as the activation function. Finally, the output layer has a softmax activation function which will calculate the probability for each word in the vocabulary being the next predicted word in the rhyme. A note is that I had to return a size of vocab +2 instead of vocab size. For some reason, there are 2 additional words in the dimensions of my y_train and y_test than in the vocab_size. To compile the model, I used categorical_crossentropy for the loss, adam as the optimizer, and accuracy for the performance metric. The model was then fit to the training data and tested on the test set. In order to generate the rhyme with 30 lines, each with 20 words, I created a method called “gen_rhyme”. It loops through two for loops, one for the number of words and another for the number on lines (specified when the method is called). It pre processes the input text by tokenizing and padding it and then it uses the model to predict the next word. The token of the associated probability is stored and it runs through the vocabulary to see if there is a match. If there is, this word is stored and added to the output. After each line, the method takes in a new input, consisting of the previous output minus the last word.

Results
The model was pretty successful on the training data. After 30 epochs, it had over 80% accuracy and a very small loss of just 0.60. However, the model did not do nearly as well at producing the next word for the rhyme on the test set of data. The validation accuracy was only around 22% after the 30th epoch and the loss was still pretty high at 16.55. However, the model did improve its prediction accuracy significantly over the 30 epochs, going from around 3% accuracy in the first epoch to over 20. However, looking at the graphs, it is interesting to note that while the accuracy trends upwards for both the train and validation data, the test validation increases over time. This can potentially be explained by the output of the model. The first three lines (each containing 20 words) look like completely random sequences of words. While the later lines don't become more coherent, you can see that they start to repeat themselves which can probably contribute to the increasing validation loss. The words in each line aren't identical, but they typically start with the same few words and only a few words and then there will be some variation towards the end of the sequence. The model is repeating the same few lines over and over, maybe due to the fact that these sequences have the highest accuracy. These could be the more popular words that are found in the rhymes. Also, I only provided a small input for the model to predict on which could have been too little information for an accurate prediction.





Conclusion

Overall, recurrent neural networks are a highly effective way to process information and learn about data that has a sequential nature. LSTMs (Long Short Term Memory) add an extra layer of complexity and protection against the faults of simple RNN’s that allow our model to remember past or forward looking information over a long term. It can improve the power and effectiveness of a model. As seen in the BBC News classification problem, we were able to achieve over 90% accuracy for both train and test data using an LSTM. As seen by the lower RMSE and the nearly perfectly fit predict graph, it was also very effective at predicting airline passengers using and LSTM and a look back window. Finally, while the nursery rhymes model wasn’t able to generate sequences up to the standard of a rhyme you'd sing to your child, it was still able to make a best-effort prediction of a rhyme based on a small snippet of input text.