UNITED STATES MILITARY ACADEMY

HW3

CS485: SPECIAL TOPICS IN COMPUTER SCIENCE

SECTION G2

MAJ DANIEL RUIZ

By

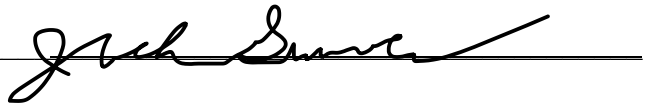
CADET JACK SUMMERS '22, CO A4

WEST POINT, NEW YORK

28 MARCH 2022

_J.S__ MY DOCUMENTATION IDENTIFIES ALL SOURCES USED AND ASSISTANCE RECEIVED IN COMPLETING THIS ASSIGNMENT.

___ _ I DID NOT USE ANY SOURCES OR ASSISTANCE REQUIRING DOCUMENTATION IN COMPLETING THIS ASSIGNMENT.

SIGNATURE: 

1. Thoroughly describe how you arrived at your final model architecture. Consider discussing where you began, how this initial baseline contributed to later decisions, and what made you finalize your architecture, rather than continue tuning?

The first step to creating the final model architecture involved investigating the starter notebook. First, I skimmed through the code to have a general understanding of the logic of the program. Then, I modified the code to upload the training set from my Google Drive. Next, I investigated how the training set was one hot encoded by printing the sequences and their corresponding next_char. Additionally, I explored the shape of the output from the one-hot encoding of both the x and y input to better understand how to form a model that would accept the information. Then I began to define the model. At first, I attempted to use bidirectional layers with an embedding layer, but I could not get it to work. In turn, I decided to implement a simple model. I added two LSTM recurrent neural network layers and two dense layers, one with 128 neurons, and one with 27 neurons for the 27 classes (a-z, "). For the first LSTM layer, I added "return_sequences = True" because there was a second LSTM layer following it. Finally, I utilized the shape method to provide the first LSTM layer with the correct expectation for the input shape.

Moving on from the model itself, I also added Early Stopping as a third callback function to help save time when training. Finally, when training, I used 1000 epochs to give it plenty of iterations to optimize and used 20 steps per epoch. I experimented with the step size and could not determine an optimal size, so I left it with 20. Finally, I used categorical_crossentropy as the loss function because the problem is multi-class in nature, and I saved the model as a .h5 model.

2. At what point did you begin to overfit on training data? What steps did you take to mitigate this and continue training?

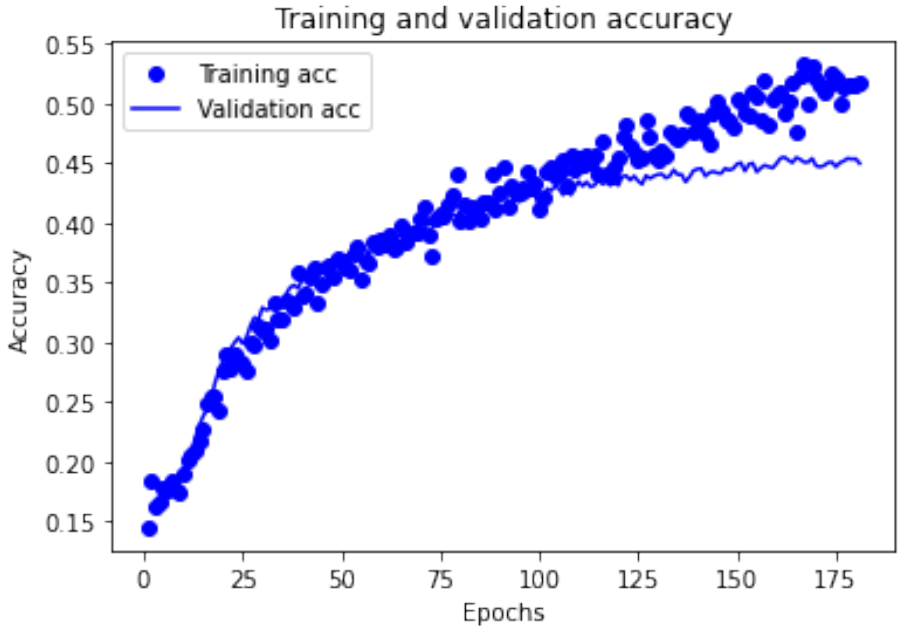
I began to overfit after approximately epoch 100. In turn, I added a dropout layer to reduce overfitting. This extended the training past epoch 150 until it started to overfit again. Additionally, the early stopping callback function terminates the training once the validation accuracy stops improving; thus the callback function helps prevent overfitting.

3. Describe what hyperparameters you modified and how did they contribute to your finalized architecture?

The primary hyperparameters I modified were the layer type, number of layers, and number of neurons. I first attempted to use an embedding layer with multiple bidirectional layers, followed by dense layers. After a few hours of modifying the embedding layer's input shape, input size, input dimensions, and output dimensions, I decided to find a more straightforward way to make the model (KISS- keep it simple stupid). I could not determine why the embedding and bidirectional layers would not accept the training data as input. In turn, I used LSTM layers and dense layers. The model began to overfit, so I added a dropout layer. Then the validation accuracy and training accuracy correlated firmly well past 100 epochs. In turn, I decided to increase the model's size in an attempt to optimize it.

4. What did you find most challenging about this process? Were there any major bottlenecks that prevented you from training further?

The most challenging aspect of the process was figuring out how to input the data into the model. I understand how bidirectional, embedding, and LSTM layers work at a high level. However, figuring out how to correctly input the data took time. The only problem that prevented further training was overfitting. The early stopping callback I added would stop the training after approximately 175 epochs due to overfitting.



5. Consider the corpus used for this problem. Is there anything about this dataset in particular that makes it easier or harder for a neural network to learn? How do you think your model would fare against other text-based corpora, and why?

I believe that URLs and hashtags (e.g., #GOARMY) may have poisoned the dataset. The model attempts to "learn" how to create coherent sentences one character at a time. In turn, strings that are not words, such as HTTPS, special symbols, or odd hashtags, made it difficult for the model to produce strings of characters that make sense to a human reader. I think my model would perform slightly better on other datasets. A 47% validation accuracy is relatively impressive for predicting between 1 of 27 classes. If other text-based corpora were input into the model, and all of the text was produced by the same author, I believe the model's accuracy would improve by a few percentage points. I think the model would improve if there were no URLs or hashtags to poison the dataset.

Example output:

"The polls that mas repustreman trump fayis falling for peceice of pascbite sayminst president gved amp orday you butmers sade nalddy what redeld bnict lowery."

Bibliography

[1] https://www.facebook.com/jason.brownlee.39, “Text Generation With LSTM Recurrent Neural Networks in Python with Keras,” *Machine Learning Mastery*, Aug. 03, 2016. https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/.