### ###Char-based LSTM model

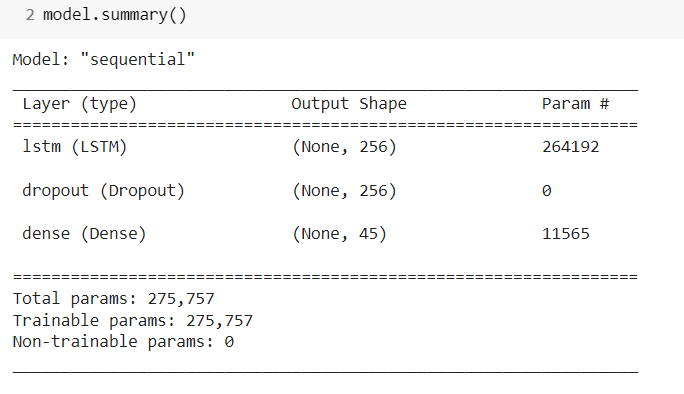
I introduced a very simple model with only two layers. Yet, the training took hours every time. The predictions got better every time the number of epochs increased but that also increased the training time that took more than 6 hours with 100 epochs.

The critical part in this model is to prepare the data. We can present the characters in different ways. In my case, using number representation for each character. For my future improvement, I intend to present each char in one-hot vector presentation.

The model give us a rhythmic output but meaningless and irredible words.

The accuracy has improved in a significant way when training with 150 epochs.

As mentioned in the thesis ‘’There are many spelling mistakes in the dataset which result in a higher level of out-of-vocabulary words and provide noise to the learner.’’’ this amplifies the problem with the char based model.

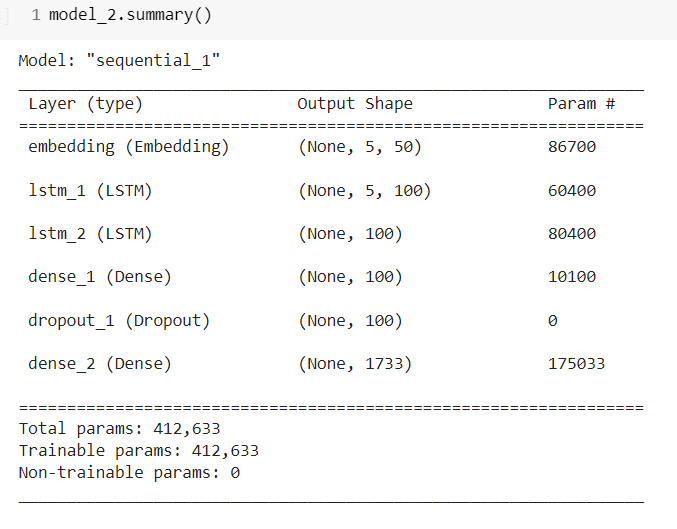


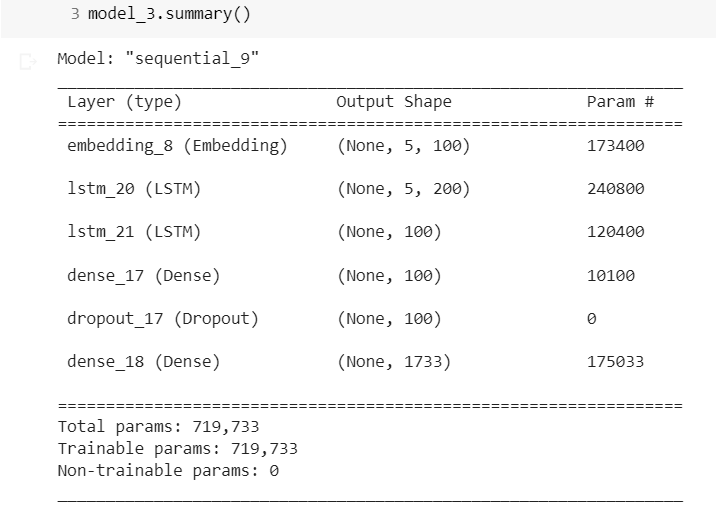
### ###Word-based LSTM model

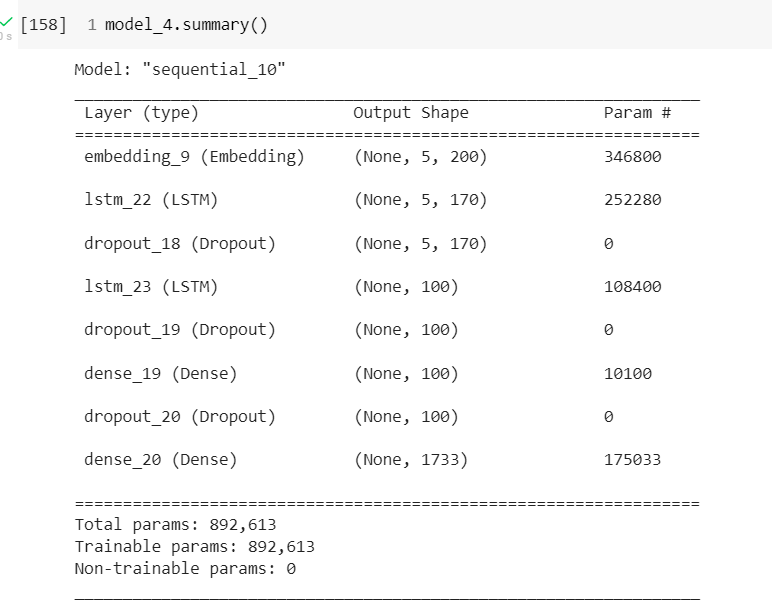
Here I choose to use an embedding layer in order to give the model the semantic presentation of each word. The embedding layer is responsible for presenting each word in a vector in an N dimensional space.

We have a very tiny vocabulary, so each time I choose a small space dimension. I ended up fixing it to 5.

Here we present the different proposed architecture.

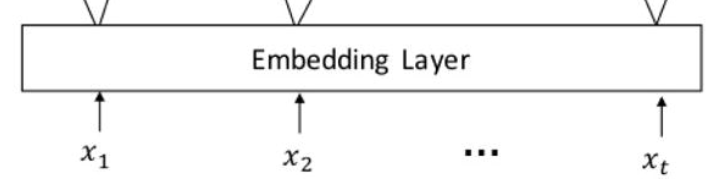






This model is composed of 5 layers:

Embedding layer that is initiated with the vocab size, input size and output size and other parameters. I chose a small embedding space of 5 dimensions because we already have a small vocabulary. The output a this layer is a 5\*200 matrix. Then, we introduce the first LSTM layer with a total of 170 units



followed bya dropout regularization

followed by another LSTM layer with 100 units

followed by a regularization

followed by a dense layer with 100 units

followed by a dropout regularization

we end up by introducing a dense layer with the vocab\_size units which in our case 1733.

I used a categorical cross entropy for all the models because the model can be simulated as a multi class classification task.

### ###Parameters Tuning

I did two trials with different optimizers Adam and RMSprop with the same architecture to see the importance of the optimizer on the model performance. The results were better using the RMSprop optimizer.

At the same time, I tried the same architecture with different units number and different dropout rates, the results were approximately similar.

Yet, the deeper proposed architecture (with more layers) has better prediction compared to others.

The number of epochs was the most important parameter to tune in all architectures. I tried a different number of epochs ( I fixed an early stopping so that if the model doesn’t improve it stopped). The bigger the number of epochs is, the better the predictions.

This parameter was really important for all architectures.

NB: It takes a really very long time to run each model each time, in a way that I couldn’t run all the models in one day and illustrate the results of each number of epochs, so the notebook contains the results of neural networks with 40/100 epochs only.

The embedding space dimension didn’t change the model prediction a lot. I think that’s due to the fact that we have a tiny vocabulary. I tried different embedding dimensions and I finally fixed the dimension to 5.

### ###Discussion

The thesis mentioned that ‘’A poetry generation system, including a person with a pen and paper, does not need to be interpretable. It is possible to enjoy poetry agnostic to the process that produced it, and read any poem at face-value. However, poetic intent can be important to the art, especially within movements like conceptual poets where the “concept of the work supplants the content of the work” and call for poems to be judged on their intents over their realization. Currently, no poetry systems are able to explain their choices in the same way that a human poet would.’’’ It’s kind of saying that it is possible to focus on the rhythms without focusing on the word.

The thesis discussed some aspects and ideas, it proposed and implemented a code that is based on LSTM that is very interesting. It proposed a human evaluation system inspired by Alan tuning experiments and some metrics like the entropy.

Markov chain is still one of the main models that generates correct lyrics that does not necessarily satisfy the rhythm, yet the markov chain idea is always discussed even for rnns.

LSTM results can be very improved depending on the different parameters that we choose to use. For the poetry generation task the real challenge is the data itself.

I end this discussion by citing this little text from the thesis ‘ Datasets which are smaller, noisier, and higher dimension are in general more difficult to learn’

The data itself is a challenge for this project as it contains some words that don’t really exist -very noisy data- so even by optimizing the different models and proposing some solutions. Yet, the problem in using transfer learning, and adding some extra words to the corpus of the singer make it even noiser data.

The idea of ‘Phoneme-level models’ cited in the thesis was genius. I used this idea in the architecture that I proposed which is a combination between markov chain and char based model.

The main added value in this project is configuring different implementations of LSTMs while trying to tune different hyper-parameters. Proposing alternatives and model conception.

The amazing thesis : Generating Rhyming Poetry Using LSTM Recurrent Neural Networks

https://dspace.library.uvic.ca/bitstream/handle/1828/10801/Peterson\_Cole\_MSc\_2019.pdf?sequence=3&isAllowed=y