Synthetic Text Generation - Progress Report

Jimmy Leroux, Frédéric Boileau, Nicolas Laliberté

27th November 2018

Abstract

In our original project plan we had indicated that we wished to explore different network topologies to build a generative model for time series data, more specifically short text sequences. After parsing through the latest literature we have opted for a Long-Short Term Memory (LSTM) network. They are several reasons for this. The most simple option would have been to use a Hidden Markov Model (HMM), however this topology suffers from several drawbacks, first the Markovian independence assumption is unreasonably strong for most text processing and using an N'th order HMM is not an alternative as time complexity grows exponentially with N. The Recurrent Neural Network (RNN) type of neural network allows one to introduce memory into an Artifical Neural Network (ANN) while keeping the model tractable. However RNN's suffer from numerical issues, most notably the vanishing-exploding gradient problem which is adressed by using using the LSTM configuration.

We have decided to build a quote generator with LSTM architecture, built using the Pytorch library, the later being state of the art in many domains, intuitive, and providing easy access to GPU acceleration. The dataset we used for the training is from Kaggle and is composed of 36.2k quotes [1]. First, we build our dictionary by extracting all the different words from our dataset. We then create an embedding layer which is responsible to learn a vector representation for each of our words. We have integrated the possibility to initialize our embedding layer with a pretrained GloVe model if we which to [2]. We are feeding sequences of 4096 encoded words to 4096 LSTM units with a hidden state of 100 dimensions. The output of each LSTM unit is then fed in a hidden layer with RELU activation functions and finally passing through a softmax activation for the identification of the predictions at the output. We are using the *cross-entropy loss as objective to optimize*, thus maximizing the loglikelihood of the training data. For the optimization technique, we *used the Adam optimizer*, since it seems to be the state of the art now [3]. To generate text, we initialize the network with a seed word, in our case it was 'What', and then feed back the predictions in the network in loops. To incorporate randomness in the generated text, we choose the predicted word by defining a multinomial distribution on the output of the softmax layer. Here's an exemple of generated text: "What is the best thing that has arises from me is coming out of mental exaltation."

In the following weeks we plan to explore different ways to make the model perform better. Now that we are confident in the general configuration of our model (LSTM) we can experiment with the cost function, the activation functions (which may differ per layer) the depth of the model, the number of hidden units and the general optimization procedure. Even a choice of optimization procedure leads to other questions for tuning our model as they are themselves governed by hyperparameter. If time allows we will also compare the main options for the vector encoding of words, the two contenders being Glove2Vec and word2vec.
Finally we will consider other tasks than the generation of synthetic text and evaluate what changes would need to be performed to tune it for those other NLP applications.

References

[1] Quotes dataset. https://www.kaggle.com/coolcoder22/quotes-dataset/home.

[2] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.