**MSDS 458: Text Generation with Bidirectional LSTM**

**6/13/2020**

**James Salafatinos**

**Abstract**

In this paper, we look to explore text generation and understand the necessary components to modeling a sequential prediction task. We aim to create a model capable of hallucinating in the style and verbiage of a body of text for representative, but novel sampling with stochasticity. We compare common memory based recurrent models in a micro-training set due to computational and time limitations but explore training one of the models on significantly more text over 10 hours to achieve interpretable results.

**Introduction**

Generation models have a wide range of applications including autonomous authoring of data driven financial reports, generalized chatbots, and as an “idea generator” companioned to our human creative efforts. We may refer to text generation is a subclass of artificial intelligence and computational linguistics to produce novel textual data from input text data. However, text generation may not be limited to language-based input data but could also be representative of other data converted into text.

**Literature Review**

In 2015, Andrej Karpathy released a legendary blog post that articulated the power of recurrent neural networks and LSTMs for text generation. This post sparked a democratization in the code available to write text generation models as it was simple and coherent [1]. In 2018, a survey was performed on the research landscape of language models for text generation; they are a part of a class of models called ‘Natural Language Generation” (NLG) [2]. In 2020, a group of researchers set out to reach the limits of our current state of the art language models. Using an evolved version of LSTM and GRU language model architecture (Transformer), researchers successfully trained a 175 billion parameter model. This model performed excitingly well on a number of tasks including text generation in which 52% of human evaluators could not tell the generated articles from real articles [3].

**Methods**

The dataset explored was an excerpt from the bible for the model comparisons, and the whole bible for the full 1-time training event to produce interpretable and coherent textual hallucinations. We explore several memory-cell based recurrent neural network models on the NLG task of “next word prediction.” GRU, LSTM, and BiDirectional LSTM are compared to understand their complexity capabilities and performance on micro-text sets Each model used encapsulated 128 memory cells to be trained with representing the model’s complexity capability. Both the GRU and the LSTM will use one layer; the BiDirectional LSTM is implemented as a two-layer LSTM with the training data reversed in order to give the model a more generalized conception of word prediction. The GRU and LSTM models are comparable in their applications, but the GRU boasts faster training times and similar performance vs the LSTM. The models explored are in the context of a sequence to vector prediction task using the softmax function to distribute the probability of each word in the corpus over the next word. To generate predictions, we use the softmax probability distribution to then sample with stochasticity to create novel prediction results after training. Each model used (GRU, LSTM, BiDirectional LSTM) encapsulated 128 memory cells to be trained with representing the model’s complexity capability. As for evaluation metrics and loss functions, we use categorical cross entropy to drive our model’s early stopping, and analysis. The loss function will be applied across the softmax vector with a size equal to the entire vocabulary encapsulated in the training and testing set. After initial experiments, we propose a 256-memory cell BiDirectional LSTM to be trained on the entirety of the bible over 100 epochs.

**Results**

The challenges of this project mostly revolved around

* + Data preprocessing
  + Memory requirements
  + Compute time requirements

Preprocessing the data turned out to be the task that requires the most code and most time in the modeling process. From gathering the data, to shaping and sculpting it into model-readable form, it required many hours of creativity and problem solving. Linked within preprocessing, was the task of converting the truth vector (i.e. the next word in the sequence) into a one hot encoded vector in the scope of the entire vocabulary. This task, naively implemented, requires 164 gigabytes of RAM to execute on the full text file of the bible (42mb). Instead, a custom generator that encodes and one-hot vector on the fly was used to feed data into the model with a batch size of 512. Not only did the NLG task have significant memory requirements, but even with the generator implemented, the time to complete 15/100 epochs using early stopping with the full 42mb of the bible required over 10 hours of learning on a single modest GPU. Google Colab and its free compute services could not be used for this task as the kernel could not stay alive for the duration of the training. In the preliminary testing stages comparing the GRU, LSTM and BiDirectional LSTM. The models appear similar in underfitted performance. However, in Figure 1, we see that the BiDirectional LSTM has more room to run on its loss function over 8 epochs compared to the other models. The final model performed almost 200% better than any in the preliminary testing most likely due to the number of training samples as the preliminary models were severely underfit.

**Final Predictions:**

The model when undertrained with the micro-data set had trouble bringing in the full vocabulary to its predictions. It also could not capture the complexity required to generate semantically correct sentences. This means that a significant amount of training data is required to engineer a useful generation model.

*"and when the woman saw that the tree was good"*

*and when the woman saw that the tree was good multiply toward it egypt, famine man man they called thing egypt. bury then wives his no lord peradventure saying, angel them, you earth, city ephron laban seed shechem day. gave taken eyes abraham thee. die. young thy face the keep years: lord, into master lord daughters, speak isaac whose years:*

But trained on the full bible text (42mb) we achieve excellent results using 512 units in a BiDirectional LSTM. The generation mechanism significantly grows in competency from the first epoch to the 10th.

First Epoch:

***"3:14 but continue thou in the things which thou hast"***

*3:14 but continue thou in the things which thou hast*

*the*

*of the and of the*

*and of the*

*of the and and the lord and the*

*10th Epoch:*

***"lo, he doth send out his voice, and that a"***

*lo, he doth send out his voice, and that a*

*man's hand shall be like to him, that is the*

*everlasting life with him.*

Exceptional. We’ve used human favorability evaluation as the context of an NLG may be used as an “idea generation” mechanism. However, there does exist intrinsic and extrinsic methods of evaluating generation performance like BLEU. It appears that 30% accuracy is enough to produce quality generated text, if using sample stochasticity.

**Future work:**

For future work, I plan to compare the word embedding similarity between religious text. With a learned language model, how does the model represent the word embeddings for Jesus and Kindness and Goodness and Light and Dark and Evil (archetypal words), when trained on various religious texts? How do the embeddings compare to existing embeddings like, GloVe?

**Conclusions**

NLG with a BiDriectional LSTM requires significant computational resources to train and to produce appropriate human evaluated results. However, this method can generate plausible hallucinations of sentences given a seed. This method can be applied into many domains of research and industry applications; one area could be lyric generation for musical artists looking for some boosted ideas.

**References and consulted sources**

Bible. (n.d.). Retrieved June 13, 2020, from http://www.gutenberg.org/cache/epub/621/pg621.txt

Understanding LSTM Networks. (n.d.). Retrieved June 13, 2020, from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Brownlee, J. (2019, August 05). Multivariate Time Series Forecasting with LSTMs in Keras. Retrieved June 13, 2020, from https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/

Using generators in Python to train machine learning models. (2018, October 13). Retrieved June 13, 2020, from https://www.jessicayung.com/using-generators-in-python-to-train-machine-learning-models/

[2005.14165] Language Models are Few-Shot Learners. (n.d.). Retrieved June 13, 2020, from https://arxiv.org/abs/2005.14165

The Unreasonable Effectiveness of Recurrent Neural Networks. (n.d.). Retrieved June 13, 2020, from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[1703.09902] Survey of the State of the Art in ... - arxiv.org. (n.d.). Retrieved June 13, 2020, from https://arxiv.org/abs/1703.09902

Fig 1 Training / Validation accuracy for preliminary testing

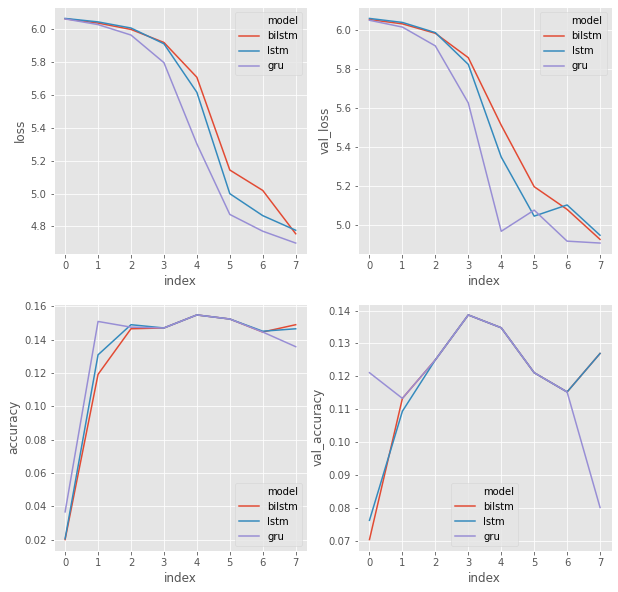


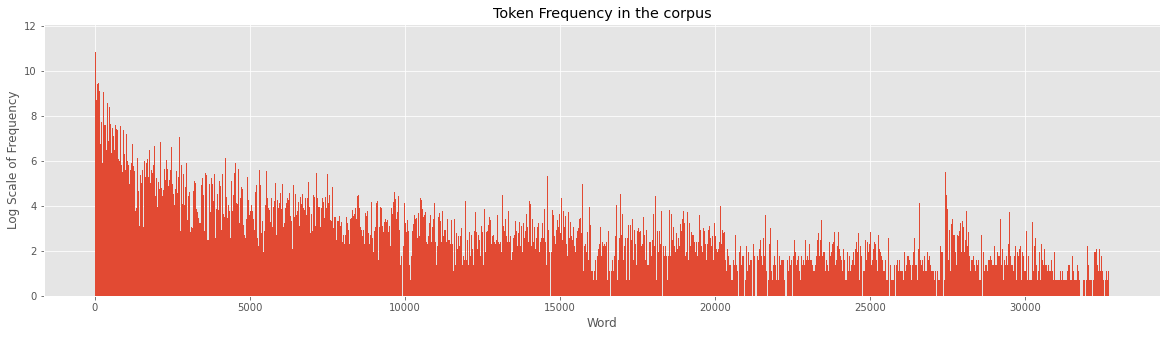
Fig 2 The long tail of vocabulary in text

Fig 3 Training and validation Accuracy and Loss for the final model

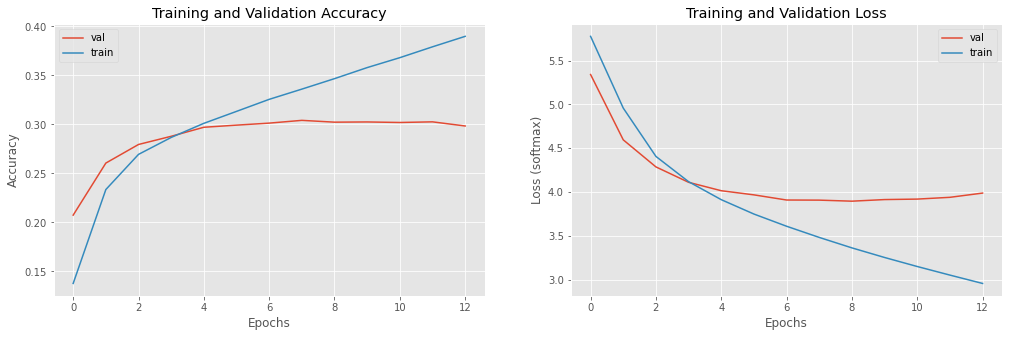


Fig 4: Weights of BiDirectional LSTM

