Final NLU project

Dinh Dung, Van (229717)

University of Trento

dinhdung.van@studenti.unitn.it

1. Abstract

This document is the report for the final Natural Language Understanding course project. The goal of this project is to implement a Language Model using one of the recurrent network architectures (Vanilla, LSTM, GRU) and evaluate on Penn Tree bank dataset.

2. Introduction

Language Model can be considered the core component of Natural Language Processing system, as it learns the contextual information through word order which helps to improve other NLP tasks such as Machine Translation or Speak Recognition. Given sentences, Language Model can extract the context provided and determine the probability of the next word sequence. This report will represent my implementation of a Language Model using Long-Short term memory (LSTM) architecture, which was developed from scratch but follows the idea of [1] and [2]. The project compares the result of designed components to enhance the baseline LSTM architecture including: drop-out, effective-history, shared weight between encoding and decoding components, and lastly a modified LSTM. Due to the limitation in acceleration hardware, the result was obtained by training on Google Colaboratory for only a limited time. The sections of the paper are structured as follows. Firstly, the Data Description Analysis will explain the statistical measurement, preprocessing procedure and label generation. Secondly, the implementation of model architecture and components will be described in the Model part. The training procedure and result will be discussed in the Evaluation part, followed by a comparison with the baseline method.

3. Data Description Analysis

The data used for this project is word-level Penn State bank dataset, which contains three subsets: trainset, validset and testset. This dataset contains 10000 unique token, which do not includes any capital letters, or punctuations. Any numbers in the dataset were changed to $'N'$ token and any name was replaced by $' < ukn >'$ token.

The training set contains 42068 sentences with 10000 unique token. The validation set contains 3370 sentences, 6021 unique tokens with no Out of Vocabulary token. The test set contains 3761 sentences, 6048 unique tokens with no Out of Vocabulary token.

For the data preprocessing, the data was taken at word-level by splitting blank space between words. A $' < start >'$ and $' < end >'$ were added to the beginning and ending of each sentence. Any word Out of Vocabulary will be replaced by $' < unk >'$ token, while $' < pad >'$ token will be added to create batch training, which increases the training speed and will not be considered during the evaluation. The label of each dataset will be the consecutive word given a previous word sequence.

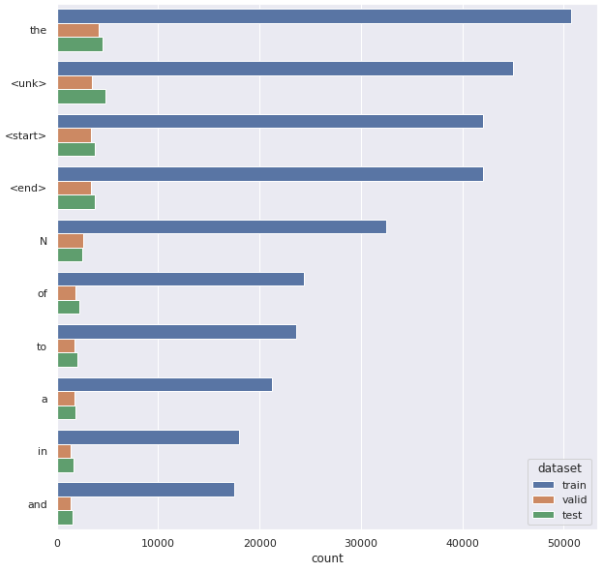

Figure 1: *Word count of the top 10 frequent words in dataset*

4. Model

4.1. Baseline model and regularization techniques

The architectural baseline consists of the following components: the encoder using Embedding, LSTM single-channel, and decoding using a fully connected layer. Words in the dataset will be converted into the correspondence index of the embedding. These embedding features will be consecutively put into the LSTM channel, the output of which will be put into the fully connected layer to predict the probability of the next word over 1000 unique tokens in the training dataset.

Regularization techniques used in the architectural baseline include:

- Drop-out: Based on the idea retrieved from [2], the researcher uses drop-out on the embedding layer, hidden layer and activation layer to increase the accuracy of the model.
- Effective history: Based on the idea retrieved from [1], the researcher uses a text to extract contextual information, in order to increase the accuracy of the next prediction. Instead of using recurrent neural networks, the researcher use the convolution neural network with a dilated layer to increase the receptive field of the model, yet the idea can be used to make a comparison with the previous output of the LSTM channel.
- Shared weight between encoding and decoding layers: The researcher uses the idea retrieved from [3] to decrease the number of training parameters to speed up the

training.

- Backpropagation Through Time (BPTT): The researcher uses the idea retrieved from **??** to improve the quality of the model.

4.2. Design of model architecture based on global and local context

Apart from those techniques mentioned above, I have tried modify the baseline architectural to compare the results. Based on the designed of LSTM, the output layer would represent the long-term memory and the hidden layer would represent the short-term memory. Every time the parameter training is updated for the prediction of the next word's probability, the context on the output layer will be adjusted accordingly based on the forget gate for the previous output signal and the update gate for the current input signal. The information will be again feed in the next prediction, which leads to the information on the output layer only suitable to predict words in a short period, causing missing information presented long before. Based on the principle that the global context (the whole idea of the text) will contribute to the prediction of the coming word in the current sentence (the local context) and the global context should not be affected by the uncompleted local context, the architecture has been modified to ensure that these two contexts are updated simultaneously.

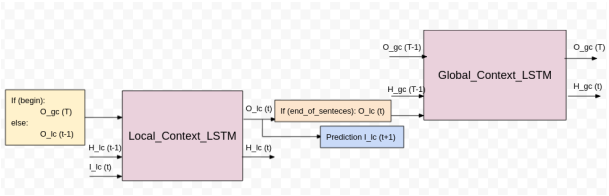

Figure 2: *The update procedure for global and local context LSTM*

The global and local context will be learn from two separate LSTM channels. The LSTM channel for the local context will be initialized at every beginning of the sentence, based on the global context presented previously (the data in the previous output channel) and the current sequence of words to predict the next word in the whole sentence. The LSTM channel for the global context will be updated using the output of the local context LSTM channel, after the whole sentence has been observed comprehensively. By this way, the context of the whole text will not be affected by partially observed information from local context. The architecture is demonstrated in Figure 2

5. Evaluation

The metrics used in this evaluation is Perplexity (PPL). I did the train and validation on sequence ranging from 100 to 200 words and test on sequence of 300 words for each batch. Based on the result, the baseline model with no modification achieved 92 PPL, while the model with added dropout achieved 80.4 PPL. Result of the evaluation is presented in figure 3

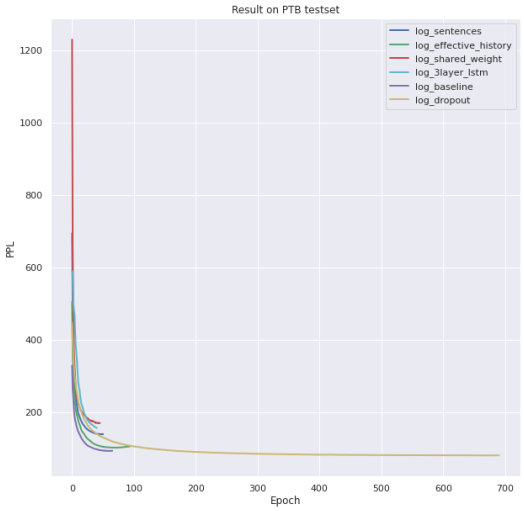

Figure 3: *Result on testset with difference setting*

Because of the limitation in acceleration hardware, some training only can run for 60-80 epochs, which will be visualized in Figure 4

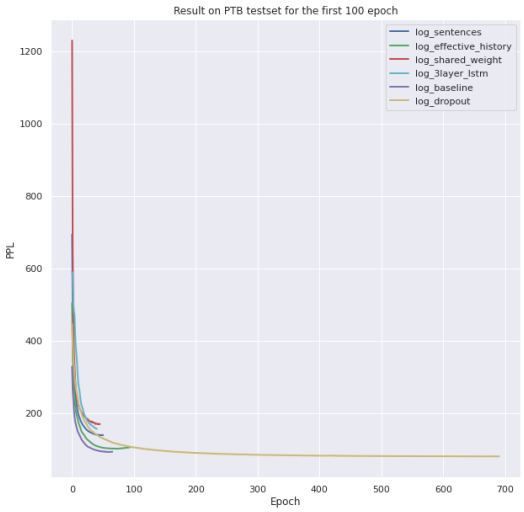

Figure 4: *Result on testset with difference setting for the first 100 epoch*

For other model setting, the effective history start to converged very soon, showing low capability in learning the context information. While the sentences setting (self-designed global local architecture) and setting with 3 layer LSTM have lower convergence rate due to high number of parameters. For the case of global and local context model, due to limitation in acceleration hardware, global LSTM model only be trained at max 4 consecutive sentences, which may affect the result

6. Conclusion

The report represented my work for the course Natural Language Understanding. Based on the experimental result, the setting with drop-out produce the best effect for the performance, while the effect history and shared weight shows less or no improvement compared with baseline model.

7. References

[1] Bai, Shaojie and Kolter, J. Zico and Koltun, Vladlen. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. arXiv:1803.01271v2

[2] Merity, Stephen and Keskar, Nitish Shirish and Socher, Richard. Regularizing and Optimizing LSTM Language Models. arXiv:1708.02182v1

[3] Inan, H., Khosravi, K., and Socher, R. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. arXiv preprint arXiv:1611.01462, 2016.