ORDERED NEURONS REVIEW: INTEGRATING TREE STRUCTURES INTO RECURRENT NEURAL NETWORKS

ChangKai Fu, Beltran F. Larry & Yadavalli Harika.
MSc Data Science, University of Southampton, United Kingdom
{ckf1n19,lrbf1n19,hy3u19}@soton.ac.uk

ABSTRACT

As The paper "Ordered Neurons: Integrating Tree Structures into Recurrent Neural Networks" (1) was awarded as the ICLR 2019 Best Paper, our work focus on experimenting the code by using different size of hidden layers and GPU settings. Fortunately, we achieve better results for WSJ data comparing to the original work. The performance of the reproduced work will be illustrated by loss graph and accuracy matrix.

1 INTRODUCTION

Fascinated by the pioneered idea to combine the tree structure and LSTM to form a new type of recurrent neural network, "ON-LSTM", we manage to extract the architecture of this neural network to see how exactly the "order" can be created in a tree structure. In this project, the original architectures built by the author, activation function "cumax" and the integration of input, forget and overlap gates, which are the crucial components for this paper will be analysed and illustrated by graphs. At last, language modelling and unsupervised parsing will be performed so that the comparison to the existing records can be made to see if the claims by the author can be reproduced by our environmental settings.

2 ARCHITECTURE OF ORDERED-NEURON NETWORKS

ON-LSTM cell is based on LSTM cell architecture, a new forget, and input gates have been defined, which utilise the function called cumax in order to allow self-organisation in the cell state(long term memory) resembling a tree structure.

2.1 ACTIVATION FUNCTION: CUMAX()

The cumax activation function is defined as the cumulative sum of the softmax function.

$$\widehat{g} = cumax(\cdots) = cumsum(softmax(\cdots)), \tag{1}$$

The intuition behind cumax could be explained as it is shown in figure 1. Let us use the softmax as a relaxation of argmax function, which returns one in the position of the maximum value. In the example, the output of a linear layer pass through argmax function, then the cumulative sum propagate the one to the next slots. As a result, the output is split into two sections in an order, and those characteristics are exploited by the master forget and input gates, as explained in the next section.

The actual cumax activation function is continuous and differentiable, which are excellent properties for back-propagation.

2.2 FORGET, INPUT AND OVERLAP GATES

The master forget ($\widetilde{f}_t$) and input ($\widetilde{i}_t$) gates are the basis of ON-LSTM, they utilised the cumax function to divide their output into two section.

In the case the master forget gate, intuitively speaking, one section selects what must be forgotten and other what remains intact. In figure 1, the red section is fill of zeros, which means that these positions must be forgotten.

Similarly, for the master input gate, one section selects what must be written. In figure 1 the green section is fill of ones, meaning that these positions must be written into the long term memory.

In the equations 2 and 3 is shown that they are working on the opposite way, one is monotonically increasing while the other is monotonically decreasing, it is worth to mention that the weight matrices are independent between them.

Ideally, the same section that is being forgotten should be written; otherwise, ON-LSTM takes advantage of LSTM to manage the overlap, ON-LSTM computes the overlap section as shown in the figure 1 and formally expressed in equation 4, then use the default behaviour of LSTM only over the overlap, for either forget and input gates in LSTM.

$$\widetilde{f}_t = cumax(W_{\widetilde{f}}x_+U_{\widetilde{f}}h_{t-1} + b_{\widetilde{f}}) \tag{2}$$

$$\widetilde{i}_t = 1-cumax(W_{\widetilde{i}}x_t + U_{\widetilde{i}}h_{t-1} + b_{\widetilde{i}}) \tag{3}$$

$$\omega_t = \widetilde{f}_t \odot \widetilde{i}_t \tag{4}$$

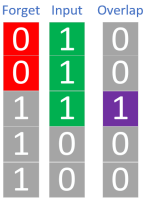

Figure 1: Intuition behind forget and input gates and their overlap. Which are the basis of the ON-LSTM

Finally, the cell state is updated as it is done is LSTM but using the ON-LSTM forget and input gates.

The figure 2 shows all the components and their connection, red lines are related with the forgetting operations, green lines with the input operations, violet lines with overlap management, grey lines do not play a vital role in the operations, for example, the section that is not going to be forgotten when the forget operation is taking place; the yellow and light green lines are the forget and input gates from LSTM respectively.

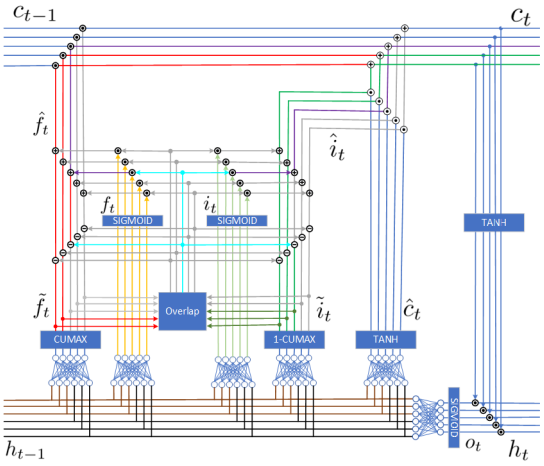

Figure 2: ON-LSTM internal cell architecture, f_t and i_t are forget and input gates from LSTM, they are been partially replaced by $\hat{f}_t$ and $\hat{i}_t$, the f_t and i_t contribution is computed dynamically based on the overlap.

3 EXPERIMENTS

The proposed ON-LSTM model claims that the ordering of the neurons helps with the language Modelling tasks by **learning the tree structures** in the **natural language** and various linguistic problems.

3.1 BRIEF SUMMARY OF THE GIVEN EXPERIMENTS:

There are four different experiments done based on the objective.

3.1.1 LANGUAGE MODELLING:

The task is to evaluate the model's ability to deal with various linguistic phenomena, e.g. co-occurrence, syntactic structure, verb-subject agreement etc. The evaluation is done by measuring perplexity with the Penn TreeBank (PTB) using splitCrossEntropy loss function to calculate the loss in the hierarchical structures.

3.1.2 UNSUPERVISED CONSTITUENCY PARSING:

This task is to compare the Latent tree structures induced by the model with human-annotated structures. Evaluation is done for each sentence (splitCrossEntropy as loss function) and the mean values across the test dataset as the total outcome.

3.1.3 TARGETED SYNTAX EVALUATION:

The task is to prove that ON-LSTM performs better on Long-term dependency (over short-term) on three structure-sensitive linguistic problems proposed in the paper Marvin Linzen (2018) [1] are subject-verb agreement, reflexive anaphora and negative polarity items.

3.1.4 LOGICAL INFERENCE :

The task is to compare the performance with the model proposed by Bowman et al. (2015) [2], when predicting the correct label (logical operation) given two pair of sentences.

3.2 REPRODUCING THE EXPERIMENTS AND IMPLEMENTATION DETAILS:

We only reproduced the experiment I and II in our work since the data and the results of experiment III and IV are not released in this paper.

3.2.1 TRAINING PHASE:

We trained the model by using the code with all the parameters the same as the author (2) except for the different size of hidden layers and the number of epochs. Code is added to generate the loss curves and save the well-trained dependencies in the model.

We trained the model using GPU(4GB) with reduced parameters, i.e. 700 hidden units instead of default 1150 quoted by the author. We have tried both ECS GPU and VM machine allocated to yann.ecs.soton.ac.uk, but the prior one is faster comparatively and also efficient as training can still be implemented even if the connection to ECS GPU is lost, but it is not the case with ssh connection. Therefore, we utilised ECS GPU to train 100 epochs instead of the default 1000 epochs in the paper since it would take about three weeks. Through rough calculation, each epoch was talking approximately 30 minutes.

3.2.2 TESTING PHASE:

Experiment I: Language Modelling
We reproduce this experiment by using Penn treebank dataset and got test loss with **4.638** after 100 epochs. **Code changes:** We write our code to train the model for 100 epochs and the train and validation curves loss is drawn. In figure 3, we could observe that the model is obviously under-fitting since the training loss is much lower than the validation one even we tried 100 epochs. The most straightforward solving method is to enhance the size of the hidden layers. However,700 hidden layers are the maximum number we can achieve in terms of the capacity of GPU we have.

Experiment II: Unsupervised Constituency Parsing
We reproduced this experiment by taking 5% of the PTB treebank from NLTK(1921 sentences). The results are not directly comparable in this case because the author took WSJ10 dataset with 7422 sentences.

[1] Rebecca Marvin and Tal Linzen. Targeted syntactic evaluation of language models. arXiv preprint arXiv:1808.09031, 2018.

[2] Samuel R Bowman, Christopher D Manning, and Christopher Potts. Tree-structured composition in neural networks without tree-structured architectures. arXiv preprint arXiv:1506.04834, 2015.

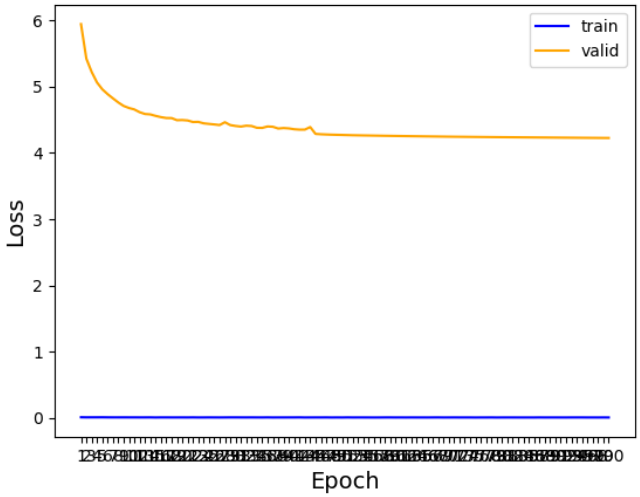

Figure 3: Train and Validation Loss after 100 epochs for experiment I, Language Modeling

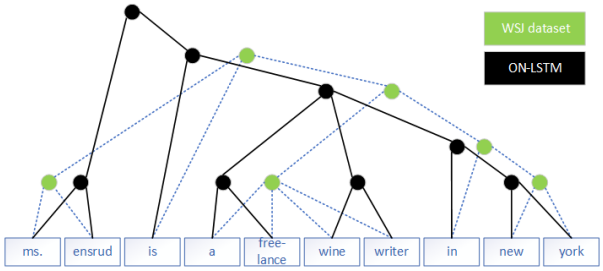

Figure 4: Hierarchy Comparison of Sample output for Experiment II, Unsupervised Constituency Parsing

Code changes: We modified the test code to download PTB treebank and test the results.
For the following sample output, we have drawn the hierarchy comparison model of the model output with the standard PTB hierarchy as shown in the figure 4.

```
          <eos> [0.91 0.79 0.21]        [0.43 0.22 0.7 ]
Standard output: [['ms.', 'ensrud'], ['is', [['a', 'free-lance', 'wine', 'writer'], ['in', ['new', 'york']]]]]
Model output: [['ms.', 'ensrud'], ['is', [[['a', 'free-lance'], ['wine', 'writer']], ['in', ['new', 'york']]]]]
Prec: 0.750000, Reca: 1.000000, F1: 0.857143
```

Figure 5: Unsupervised Parsing - Sample Output

Parsed Results comparison of the reproduced model with the results quoted in the paper given by the table.

		Accuracy Metrics			
Model	Depth WSJ	ADJP	NP	PP	INTJ
ON-LSTM Paper-3rd layer	5.3	44.8	57.5	47.2	0.0
Reproduced - Results	5.68	46.6	67.0	60.6	50.0

Table 1: Parsing results - Accuracy comparison with results quoted in the paper

4 Conclusion

As a reproducibility challenge, we have reproduced the essential claims made by the ON-LSTM paper and compared the results along with a clear explanation of the intuition behind the architecture changes to the LSTM cell. Interestingly, we have found that unsupervised parsing results are better than the accuracy results claimed in the paper; it could be because the test dataset is different.

References

[1] Shen, Yikang, Shawn Tan, Alessandro Sordoni and Aaron C. Courville. “Ordered Neurons: Integrating Tree Structures into Recurrent Neural Networks.” https://openreview.net/forum?id=B1l6qiR5F7

[2] Code in the github: https://github.com/yikangshen/Ordered-Neurons