Saarland University
Department of Computer Science
Summer term 2019
Seminar Mathematics of Deep Learning
Lecturer: Dr. Antoine Venant

Seminar Paper

LSTM as a Dynamically Computed Element-wise Weighted Sum Implementation

Name:	Sara Khan
Matriculation number:	2517648
Field of study:	Masters in Computer Science
Email:	s8sakhan@stud.uni-saarland.de
Date:	September 27, 2019

I hereby confirm that I have written the seminar paper with the title **LSTM as a Dynamically Computed Element-wise Weighted Sum Implementation** in the seminar **Mathematics of Deep Learning** (lecturer: **Dr. Antoine Venant**) on my own and that I have not used any other media or materials than the ones referred to in this seminar paper.
I further confirm that I did not submit this or a very similar seminar paper in another seminar.

Saarbrücken, September 27, 2019 _____

Contents

1 Introduction

Long-short term memory (commonly known as LSTM) neural networks are widely used for their ability to handle temporal dynamic behaviour. The paper "Long Short-Term Memory as a Dynamically Computed Element-wise Weighted Sum" [1] provides a different view to the success of LSTM. The paper states that gates themselves are powerful recurrent models that provide more representational power than previously realized. In order to prove it, the authors perform different LSTM ablations on four main applications of NLP. The paper also mathematically proves that the LSTM variant "LSTM-SRNN-OUT (Remove S-RNN and OUTPUT gate from LSTM) can be condensed into element wise weighted sum of the previous layers.

The report describes the implementation of the above titled paper for the task of word level language modelling. The implementation is done on Penn Tree Bank dataset for a fixed set of hyper-parameters. Results of the implementation are similar to the original paper. Perplexity scores of different LSTM variants clearly shows that the gating mechanism alone performs as well as an LSTM in most settings.

2 Motivation

The success of LSTMs are largely owned to their ability to combat vanishing gradient problem in Recurrent Neural Networks(RNN). Memory gates present in LSTM mitigate it. The study reveals an alternative view to the wide success of LSTM. It states that gates themselves are powerful recurrent models. Thus, it mathematically as well as empirically provides a new direction in which LSTMs can be viewed. The authors' break down the internal structure of the cell into different sub-components and ablate it. This is first study to provide comparison between LSTMs with and without recurrent layer. It focuses on explaining what LSTMs are learning. This can be very helpful for research especially from the optimisation point of view. Also, there is no publicly available implementation for the above paper. Thus, it provides immense motivation to reproduce the experiment and get more insights.

3 Experimental Setup and Results

The experiment implements the task of language modelling. It predicts the next word in the sequence. The dataset used is Pen Tree Bank dataset(PTB)[1]. It contains data for training, validation and testing. The vocabulary size is 10,000 words. The implementation is written in Python language. The most important Python libraries to be used are numpy and tensorflow. The tensorflow official tutorial is referenced [2] for implementation. The hyper-parameters used in the model are as follows.

Parameter	Description
init scale	initial scale of the weights
learning rate	initial value of the learning rate
max grad norm	maximum permissible norm of the gradient
num layers	number of LSTM layers
num steps	number of unrolled steps of LSTM
hidden size	number of LSTM units
max epoch	no. of epochs trained with the initial learning rate
max max epoch	total number of epochs for training
keep prob	probability of keeping weights in the dropout layer
lr decay	decay of the learning rate for each epoch after "max$_e$*poch*"
batch size	batch size

The implementation contains three different configurations for the above parameters. The below table defines the configurations- small, medium and large used in the implementation.

[1] `https://github.com/townie/PTB-dataset-from-Tomas-Mikolov-s-webpage/tree/master/data`

Parameter	Small Config	Medium Config	Large Config
init scale	0.1	0.05	0.04
learning rate	1.0	1.0	1.0
max grad norm	5	5	10
num layers	2	2	2
num steps	20	35	35
hidden size	200	650	1500
max epoch	4	6	14
max max epoch	4	10	55
keep prob	1.0	0.5	0.35
lr decay	0.5	0.8	1/1.15
batch size	20	20	20

Due to the slow execution time, the perplexity scores are computed only for small configuration. The test perplexity scores for various LSTM ablations are described in below table.

Ablation	Test Perplexity
LSTM	132.434
LSTM - SRNN	129.044
LSTM - GATES	655.695
LSTM - SRNN - HIDDEN	126.070
LSTM - SRNN -OUT	141.061

The results clearly show that LSTM without GATES have very low performance whereas rest of LSTMs without SRNN or HIDDEN or OUT does not differ much in performance.

4 Conclusion

This report describes the implementation of the paper titled "Long Short-Term Memory as a Dynamically Computed Element-wise Weighted Sum". It reproduces the results for the task of language modelling. Both the paper and the reproduced implementation works on the same dataset (PTB). The implementation is done on a fixed set of parameters. Thus, it shows similar results to the paper and hence, we can empirically prove that the gates are doing much more in practice than just alleviating vanishing gradients.

5 References

[1] Omer Levy Kenton Lee Nicholas FitzGerald Luke Zettlemoyer, Long Short-Term Memory as a Dynamically Computed Element-wise Weighted Sum, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, pages 732–739, Melbourne, Australia, July 15 - 20, 2018*

[2] Tensorflow tutorial: https://www.tensorflow.org/tutorials/sequences/recurrent