# Language Modeling RNN network with LSTM cells

Giovanni Lorenzini (223715)
University of Trento
Via Sommarive, 9, 38123 Povo, Trento TN
giovanni.lorenzini@studenti.unitn.it

## Abstract

*This work focuses on implementing a Recurrent Neural Network (RNN) Language Model with Long Short-Term Memory (LSTM) units. This report details the realization of a LSTM cell and a RNN network. It shows also the performance obtained, so the perplexity (PPL) on the Penn Treebank word level dataset.*

## 1. Introduction

A recurrent neural network is a class of artificial neural networks that can process variable length sequences of inputs. They are especially powerful when dealing with speech related tasks. There are a lot of variants of RNNs [4]. This particular work employs the so called LSTM version, which is basically an enhancement over the classic Elman network. Generally speaking, it is known that RNNs suffer from vanishing gradients as data from past hidden states eventually gets multiplied in cascade and their contribution gets smaller and smaller to the point that it becomes negligible. LSTMs were ideated in order to prevent such unfortunate behaviour. The basic idea is that by adding some parallel branchings along the computations, it can be possible to track what had happened in the history of the input sequence, so that it the vanishing gradient issue of the RNNs could be prevented.

This work put effort in obtaining a neural network that achieves a probability model that predicts words in a sentence. Perplexity is an evaluation metric commonly used by computational linguists in context of language models that indicates how good is the model at predicting the next words given a text sequence.

Current state-of-the-art perplexities on the Penn Treebank dataset can reach values up to 20.5 by Brown *et al*. [2], but have lots of parameters and require a model that has a complex structure, while here instead I show that even a simple neural architecture with far less parameters can achieve satisfactory results.

The report is structured as follow: the first section defines more formally the problem statement, then I proceed with an analysis of the given dataset and with the description of the models employed. Ultimately, I discuss the obtained results while underlying their strengths and limitations.

## 2. Problem statement

Language modeling is the task of predicting what word comes next. We use language models every day, look for example at the iPhone keyboard in Figure 1 or at the Bing suggestions in Figure 2.

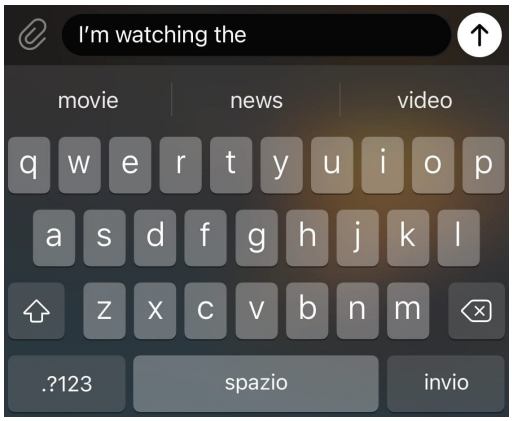


Figure 1. iPhone keyboard suggesting the next word.

More formally: given a sequence of words $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$, compute the probability distribution of the next word $x^{(t+1)}$:

$$P\left(x^{(t+1)} \middle| x^{(t)}, \ldots, x^{(1)}\right) \tag{1}$$

where $x^{(t+1)}$ can be any word in the vocabulary $V = \{w_1, \ldots, w_{|V|}\}$ [5].

To evaluate a language model is necessary to compute the perplexity (PPL), the lower the better. The perplexity is

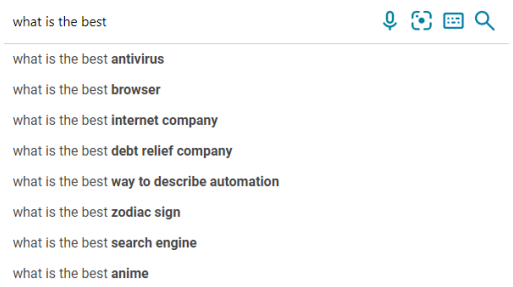


Figure 2. Bing search suggestions.

defined as:

$$PPL = \prod_{t=1}^{T} \left( \frac{1}{P_{LM}\left(x^{(t+1)} \middle| x^{(t)}, \ldots, x^{(1)}\right)} \right)^{1/T} \quad (2)$$

that is also equal to the exponential of the cross-entropy loss $J(\theta)$:

$$PPL = \prod_{t=1}^{T} \left( \frac{1}{\hat{y}^{(t)}_{x_{t+1}}} \right)^{1/T} \quad (3)$$

$$= \exp\left( \frac{1}{T} \sum_{t=1}^{T} -log\left( \hat{y}^{(t)}_{x_{t+1}} \right) \right) \quad (4)$$

$$= \exp\left( J(\theta) \right) \quad (5)$$

## 3. Data analysis

The dataset used is called Penn Treebank Dataset[1] (Taylor *et al.* [6]). It contains several english sentences, taken from manuals, journal articles, telephone conversations and others. For this work I have used only the world level dataset, so the one with the files names as: *ptb.(test|train|valid).txt*. Every line is a chunk of a sentence. The train dataset is composed of 42068 lines, while the test dataset is composed of 3761 lines. The vocabulary of the PTB dataset is capped at 10k unique words.

In this work, individual words are stored in a dictionary and sentences are put in a list. The network receives batches of sentences; each sentence is padded up to the maximum lenght of the longest phrase in the batch, to account for different lenghts. The padding element is defined as <*pad*> and will be excluded in the loss computation. More precisely, the network is fed with a sequence of words from a sentence, excluding the last one, and its output predictions have to match the ground truths which are all the elements from the corresponding sentence but the first one.

## 4. Model description

Instead of neurons, LSTM are built so that they resemble memory blocks. Their inner connections are depicted in Figure 3.

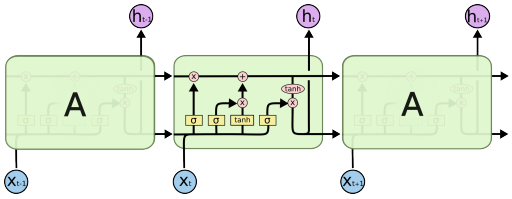


Figure 3. A single layer LSTM network.

A basic block contains four layers, each with different purposes.

First there is the *forget gate layer*, which conditionally decides what information to throw away from the cell state. Figure 4.

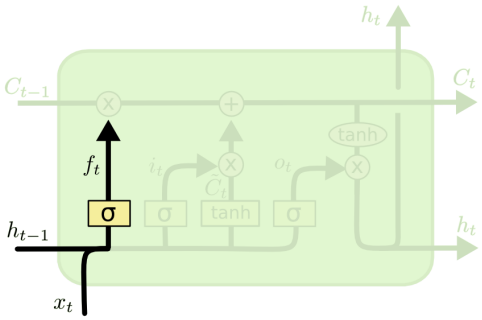


Figure 4. LSTM forget layer.

$$f_t = \sigma\left( W_f \cdot [h_{t-1}, x_t] + b_f \right) \quad (6)$$

Then comes the *input gate layer*, that decides what new information can be stored in the cell state. Figure 5.

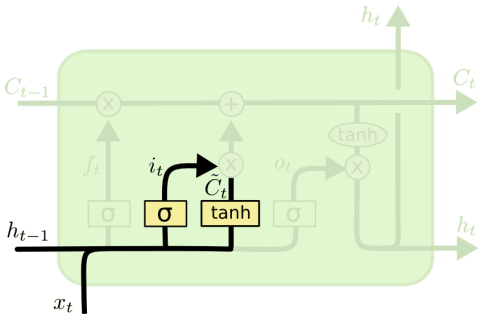


Figure 5. LSTM input layer.

$$i_t = \sigma\left( W_i \cdot [h_{t-1}, x_t] + b_i \right) \quad (7)$$

$$\tilde{C}_t = \tanh\left( W_C \cdot [h_{t-1}, x_t] + b_C \right) \quad (8)$$

Next the cell state can be updated with the decisions of the previous layers. Figure 6.

[1] https://deepai.org/dataset/penn-treebank

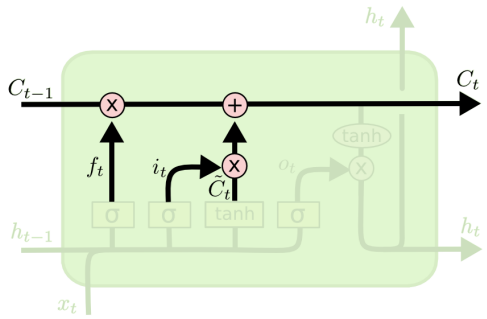


Figure 6. LSTM updating cell state.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{9}$$

Lastly, there is the *output gate layer* that decides what will be the output of the cell basing on its input and the current memory of the block. Figure 7.

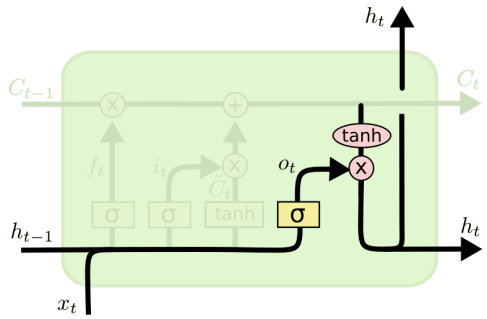


Figure 7. LSTM output layer.

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{10}$$

$$h_t = o_t * \tanh(C_t) \tag{11}$$

For each block and for each gate, there are several associated weights that will be learned during the training procedure. To obtain good performances in the language modeling task I implemented a two layers LSTM and the dropout as reccomended by Zaremba *et al.* [8]. In Figure 8 is possible to see a two layers LSTM. The dashed arrows indicate connections where dropout is applied and solid lines indicates connections where dropout is not applied.

The network I designed is based on the parameters in Table 1.

## 5. Results

Doing an hyperparameters tuning I found out that the network works well with the parameters reported in Table 1. The model has been trained on a RTX 3080 for 25 epochs taking around 2 hours. I have obtained a perplexity of 53.6.

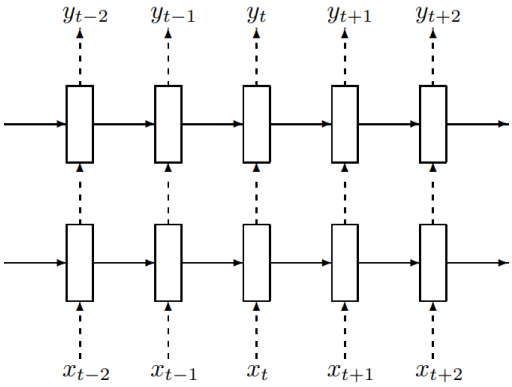


Figure 8. Two layers LSTM.

| Parameter | Value |
|---|---|
| Layers | 2 |
| Hidden size | 512 |
| Input size | 512 |
| Batch size | 64 |
| Dropout | 0.5 |
| Clip | 0.25 |

Table 1. LSTM network parameters.

In Figure 9 the training perplexity and validation perplexity are shown.

| Model | Author | Year | Test PPL |
|---|---|---|---|
| GPT-3 | Brown [2] | 2020 | 20.5 |
| BERT-Large-CAS | Wang [7] | 2019 | 31.3 |
| Mogrifier LSTM | Melis [3] | 2019 | 44.9 |
| LSTM | Zaremba *et al.* [8] | 2015 | 78.4 |
| GRU | Bai *et al.* [1] | 2018 | 92.5 |
| 2 Layer LSTM | My Model | 2021 | 53.6 |

Table 2. Results comparison.

## 6. Conclusion

The LSTM cell that I have implemented works well but at the same time is way slower than the LSTM provided by PyTorch. In fact the PyTorch LSTM is mainly coded in C with only a Python interface and the calculations are optimized.

Nonetheless, the network performances are on par of what I expected. The best word-level language model implementing LSTMs achieve 44.9 perplexity (work by Melis [3]), whereas mine stands at 53.6.

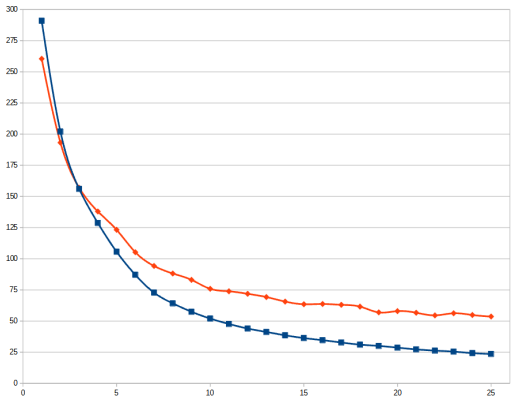


Figure 9. Training process. Blue: training perplexity. Red: validation perplexity.

## 7. Sentences generated by the LSTM

Here are some brief sentences that have been generated by the network.

*The oil market also says he was in an interview with cheaper technology investors.*

*To this model increase is a league effort for customers or three others said to be included by mr. goldberg 's being fired in june N.*

*In the country with salomon brothers inc. a <unk> <unk> has established management and public benefits.*

*In a sweetened health-care case for business would be paid to N he says an executive spokeswoman was also widely named old said an analyst with coca-cola group.*

## References

[1] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018. 3

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020. 1, 3

[3] Gábor Melis, Tomáš Kočiský, and Phil Blunsom. Mogrifier lstm, 2020. 3

[4] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. volume 2, pages 1045–1048, 01 2010. 1

[5] Abigail See. Natural language processing with deep learning, 2019. 1

[6] Ann Taylor, Mitchell Marcus, and Beatrice Santorini. The penn treebank: An overview. 01 2003. 2

[7] Chenguang Wang, Mu Li, and Alexander J. Smola. Language models with transformers, 2019. 3

[8] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization, 2015. 3