Recurrent Neural Network Language Model Implementation

Davide Lusuardi
Department of Information Engineering and Computer Science
University of Trento
Trento, Italy
davide.lusuardi@studenti.unitn.it

***Abstract*—Language models are central to many Natural Language Processing (NLP) tasks as they can provide word representation and probability indication of word sequences. Some of these tasks include Machine Translation and Speech Recognition. Recurrent Neural Network Language Models (RNNLMs) improve the performance of traditional Language Models. In this work, we show how to implement a Recurrent Neural Network Language Model using PyTorch along with the implementation of simple LSTM and GRU cells. This report also shows the performance obtained on the Penn Tree-bank dataset using the perplexity metric (PPL).**

I. Introduction

Language modeling task consists in predicting the probability of the next word. It is a crucial component in real-world applications such as Machine Translation and Automatic Speech Recognition. For example, a translation system generates multiple translations for the same sentence and the language model scores all the sentences and decides the most likely one.

More formally, Language Models assign a probability to a sequence of words: given a text corpus with a vocabulary V and a sequence of words $w_1, w_2, \ldots, w_{t-1}$, we need to compute the probability distribution of the next word w_t [1]:

$$P(w_t|w_1, w_2, \ldots, w_{t-1}) \tag{1}$$

where w_t can be any word in the vocabulary V. Language models can operate at different levels: character level, n-gram level, sentence level or paragraph level.

A recurrent neural network (RNN) is a type of artificial neural network that operates on sequences of data of variable length. In the case of RNN, the output from the previous step is fed as input to the current step. This is particularly useful in applications where there is the need to remember the previous state.

Recurrent Neural Net Language Model (RNNLM) is a type of neural network language model that exploits a RNN cell. There are several different types of RNN cells, but we will focus on Long-Short Term Memory [2] and Gated Recurrent Unit [3] cells. Since a RNN can deal with the variable length data, it is suitable for modeling data such as sentences in natural language.

II. Evaluation of language models

A. Evaluation metric

Perplexity (PPL) is the commonly used evaluation metric for language models. Perplexity is defined as [4]:

$$PPL(W) = \frac{1}{P(w_1, w_2, \ldots, w_N)} \tag{2}$$

where W is the test set. Note that a lower perplexity indicates a better model.

Perplexity can also be defined as the exponential of the cross-entropy:

$$PPL(W) = 2^{H(W)} = 2^{-\frac{1}{N} log_2 P(w_1, w_2, \ldots, w_N)} \tag{3}$$

B. Dataset

The experiments have been conducted on the Penn Tree Bank (PTB) dataset [5], which consists of 929k training words, 73k validation words, and 82k test words and a vocabulary composed of 10k words. The material annotated includes IBM computer manuals, nursing notes, Wall Street Journal articles, and transcribed telephone conversations. The dataset is available from the website *deepai.org* [6].

III. Model implementation details

In this section, we explain some implementation details of the designed RNN Language Model. The model is composed of an Embedding layer, a RNN cell and a Fully-Connected layer. The model supports two main types of RNN cell: LSTM and GRU. A simple version of these cells has been implemented.

A. LSTM cell

The structure of the Long-Short Term Memory cell is shown in Figure 1 and consists of three main gates:

1) The *input gate* decides how much input information is added to the current state and is caracterized by the following Equation:

$$i_t = \sigma\left(W_i \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_i\right) \tag{4}$$

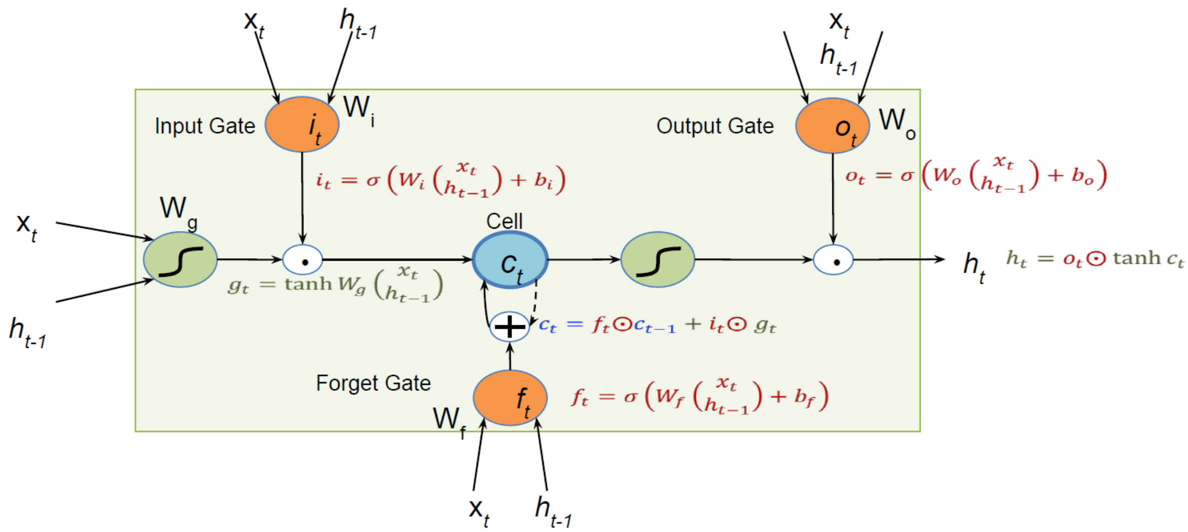

Figure 1. Long-Short Term Memory cell structure and related equations.

2) The *forget gate* decides how much of the past should be remembered and is caracterized by the following Equation:

$$f_t = \sigma\left(W_f \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f\right) \tag{5}$$

3) The *output gate* decides the output based on the current state and is caracterized by the following Equation:

$$o_t = \sigma\left(W_o \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_o\right) \tag{6}$$

The memory cell c_t is updated as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh\left(W_g \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}\right). \tag{7}$$

The new hidden state h_t is computed as follows:

$$h_t = o_t \odot tanh(c_t). \tag{8}$$

B. GRU cell

The structure of the Gated Recurrent Unit is shown in Figure 2 and consists of two main gates:

1) The *reset gate* is used to determine how much of the past information to forget and is caracterized by the following Equation:

$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_r\right) \tag{9}$$

2) The *update gate* helps the model to decide how much of the past information needs to be considered and is caracterized by the following Equation:

$$z_t = \sigma\left(W_z \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_z\right) \tag{10}$$

Finally, the new hidden state h_t is computed as follows:

$$h'_t = tanh\left(W \begin{pmatrix} x_t \\ r_t \odot h_{t-1} \end{pmatrix}\right) \tag{11}$$

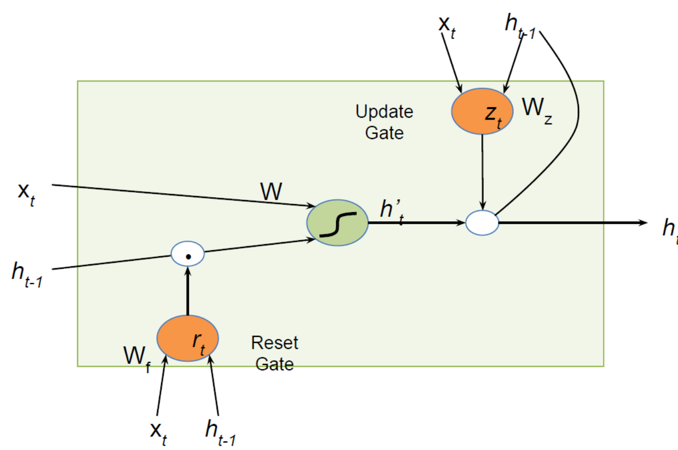

Figure 2. Gated Recurrent Unit cell structure.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h'_t \tag{12}$$

IV. Results

The model has been trained on the Penn Tree-bank training set for 40 epochs taking 1201289 hours and the hyperparameters have been tuned on the validation set. The best hyperparameters found are shown in the Table. Figure shows the training and validation perplexity over the epochs. The model has been evaluated on the test set obtaining a perplexity equal to 2184912.

References

[1] "How do language models predict the next word?" https://towardsai.net/p/nlp/how-do-language-models-predict-the-next-word.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[3] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[4] "Perplexity in language models," https://towardsdatascience.com/perplexity-in-language-models-87a196019a94.
[5] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993. [Online]. Available: https://aclanthology.org/J93-2004
[6] "Penn treebank dataset," https://data.deepai.org/ptbdataset.zip.