Exercise Sheet 10

Zena Al Khalili (7009151)
Sangeet Sagar (7009050)
{zeal00001,sasa00001}@stud.uni-saarland.de

February 2, 2021

Exercise 10.1 - Architecture

a)

RNN when fed more new data, tend to forget the data they have seen several time steps back, that is because of the problem of vanishing gradient (resulting of the multiplication of many matrices from the previous time steps). LSTM solves this problem with a more complex cell structure that can learn when to forget the information and when to save them in the long term memory.

b)

LSTM cell: Long term memory in LSTM is called cell state C_t , it consists of input gate, forget gate, output gate. The input to C_t is input x_t and the output of the previous cell output h_{t-1}. Figure 1 shows LSTM cell. The formula used to calculate each element are shown in Figure 2

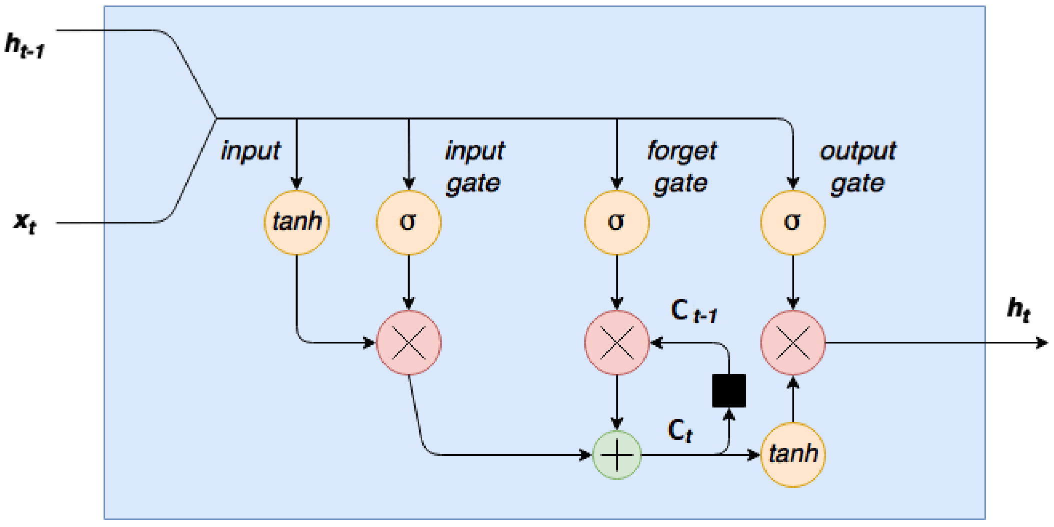

Figure 1: *LSTM Cell Structure*

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + \mathbf{b}_i)\\
\mathbf{f}_t &= \sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + \mathbf{b}_f)\\
\tilde{\mathbf{c}}_t &= \tanh(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1} + \mathbf{b}_c)\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t\\
\mathbf{o}_t &= \sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + \mathbf{b}_o)\\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}
$$

Figure 2: *Formulas of each element in LSTM cell structure*

Where:

$\tilde{c}_t$: here we apply tanh on the combined input from x_t and

C_t to scale it to be in range [-1,1], we use tanh, because we want some values to be 0 (to be forgotten) and some other values to be 1 (will be saved in memory).

i_t : here we apply sigmoid activation function on the combined input and the result would be the output of the input gate. Then multiply it with $\tilde{c}_t$ to decide which input will be saved, and which will be forgotten.

f_t : the first part of the forget gate is to apply sigmoid on the combined input with different weight matrices, if the output of the forget gate is 0 in the position of the resulting matrix the cell state will erase the memory at this position, if it's 1 the cell state will keep it.

c_t : is the sum of two elements : the first one is the dot product of input gate and the scaled input (in range[-1,1]) the second element is the dot product between the forget gate output and the previous cell state c_{t-1}.

Finally the focus vector h_t: is the dot product between tow elements: the first element: is the the combined input with sigmoid function applied to it. The second element is the cell state ct scaled to be in range [-1,1] By applying tanh on c_t

Exercise 10.2 - Embeddings

a)

A static word embedding is a function that maps each word type to a single vector. These vectors are typically dense and have much lower dimensionality that the size of the vocabulary. A good example of static word embedding is word2vec.

In contextualized word embedding we build a vector for each word conditioned on its context. Here the representation for each token is a function of the entire input sentence. A good example of contextualized word embedding is CoVe.

b)

Static word embeddings such as word2vec, generates a dictionary like embedding for each word, without taking into consideration the context in which the word appears in. So the model is fed only words to generate the embeddings.

While contextualized or dynamic embeddings, generated by models such as transformers, give a different embedding of the word with different contexts. So the model is fed the entire sentence to generate the embeddings.

E.g: - I will let you know my point of view. Where did you place the point? Static embeddings will give the same embedding for the word "point" in both sentences, while dynamic embeddings will give two different embeddings.

c)

Transfer learning is the use of a pre-trained model with fine tuning its parameters for a specific task, without training the model from scratch. We usually use transfer Learning in language tasks because it has small datasets, which makes it difficult to train Deep NNs without over fitting the small dataset. The model is pre-trained on a general Language modeling task, then fine tuned for a more specific task such as text classification.

Yes, we can pre-train without any task but this would perform terrible because it has no knowledge of the semantics of the language. This knowledge can be acquired only in the pre-training stage given a task.