A text classification using multiple-level LSTM on simulated sentences

Gonsoo Moon
Post Master Student of Computer Science
University of Texas at Dallas,
gonsoomoon@gmail.com

1. Introduction

Using multiple LSTM (Long Short Term Memory) on the simulated sentences varying in size, a network consisting of an embedding, multiple rnn, linear, and softmax layers classifies two types of sentences. A sentence is generated at the maximum of 6 in which case one less than 6 is padded with 0. The sentence has two types, one having all even numbers with [0,1] as a label and the other having all odd numbers with [1,0]. Thus, a task is to classify unseen the two types of sentence. This paper's purpose is to describe three network architectures when one, two and three level of LSTM are used each.

2. Simulated Input Data

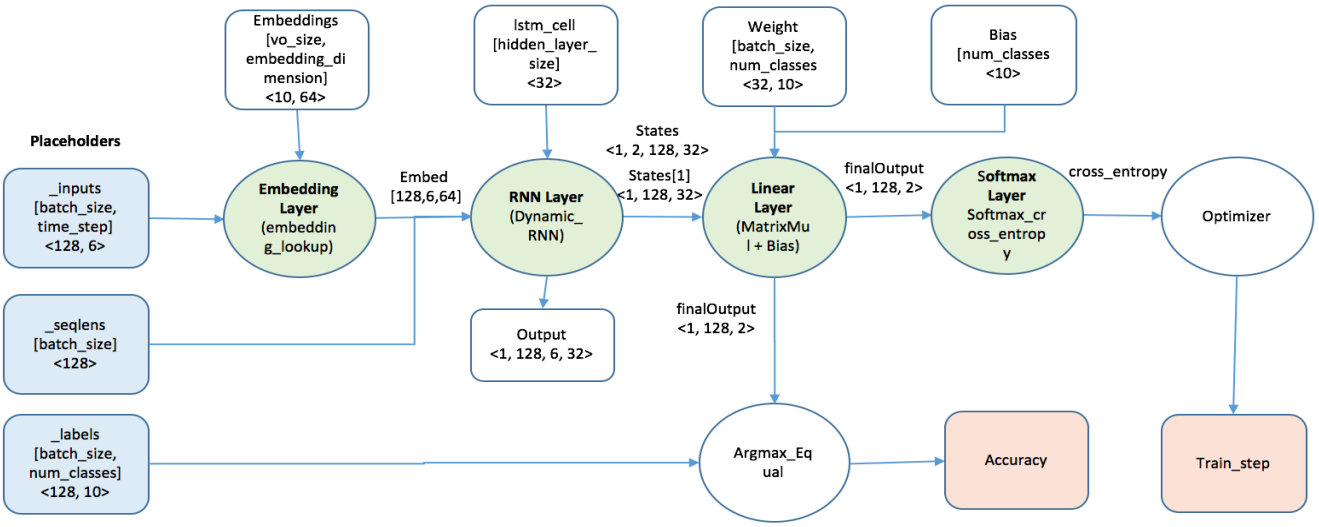
With the maximum of six words, a sentence is generated. One type of sentence having odd numbers is like [five, nine, seven, five, pad, pad] while the other type is like [four, two, eight, six, two, two]. I randomly generated 10,000 sentences each so that the total data set is 20,000. Thus, one half of the total data set was used as training and the other half was used as testing.

3. Network Architecture

<Figure1>

The main architecture consists of an embedding, RNN, Linear and Softmax layers shown in the <Figure1>. First, an input is fed into the embedding layer with a batch size of 128 and a step length of 6. Also, the embedding layer is given an initial embeddings with the embedding input shape, [10] as vocabulary size and an embedding size of 64. Through the embedding layer, the embed as a tensor with a shape of [128, 6, 64] is generated and fed into RNN layer having a sequence of six steps, each step containing 32 neurons. Also, the RNN layer is given a length of the inputs. For example, if one of the 128 batches is [two, six, two, four, eight, pad], a length is 5 in order for the RNN layer to only process five steps except for steps marked with 'pad'. The output on the sixth step as a last step among six outputs from six steps, which is named "Output"- shape of [1, 128, 6, 32] -, is the same as the states[1]. A value of the States is full of a state data and states[1] is in the States. Also, the States is in the Output. The states[1] is fed into the Linear layer playing a role in getting logits and not using non-linear activation function. The Linear layer produces a result in the form of a shape of [1, 128, 2] as the finalOutput tensor which is used for two purposes. One is to calculate a prediction calculation and the other is to calculate a gradient of a loss function, the cross entropy.

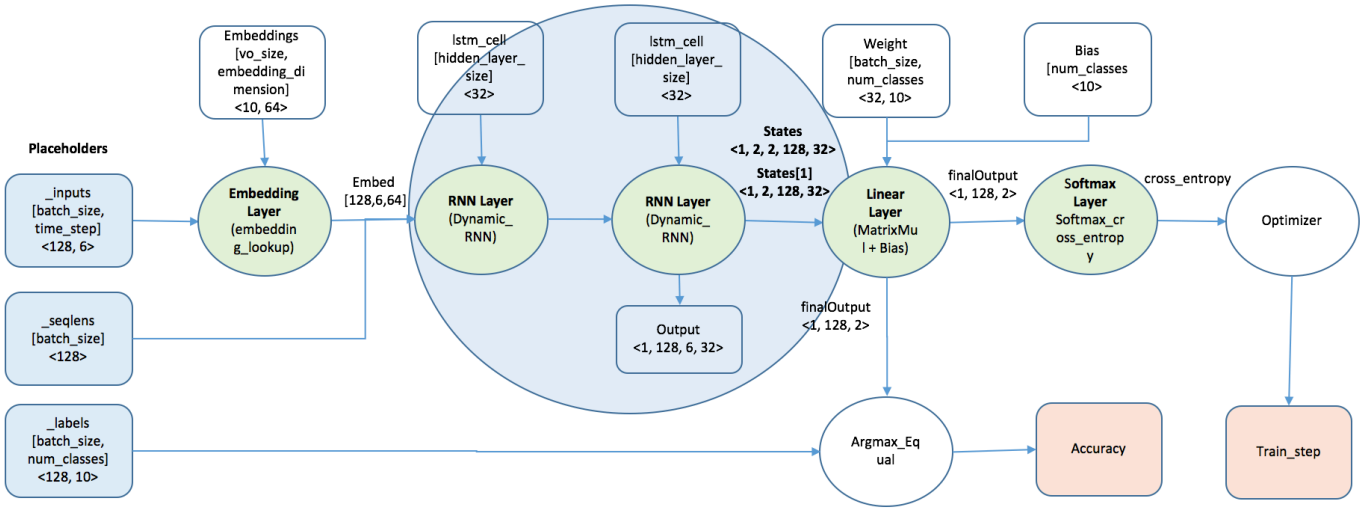
An end-to-end text classification one-level RNN Model with Embeddings



In shown in The <Figure2> case, it shows two-level RNN with all the same except for it. Specifically, the shape of the states is changed from <1, 2, 128, 32> to <1, 2, 2, 128, 32> to incorporate two stacked RNN.

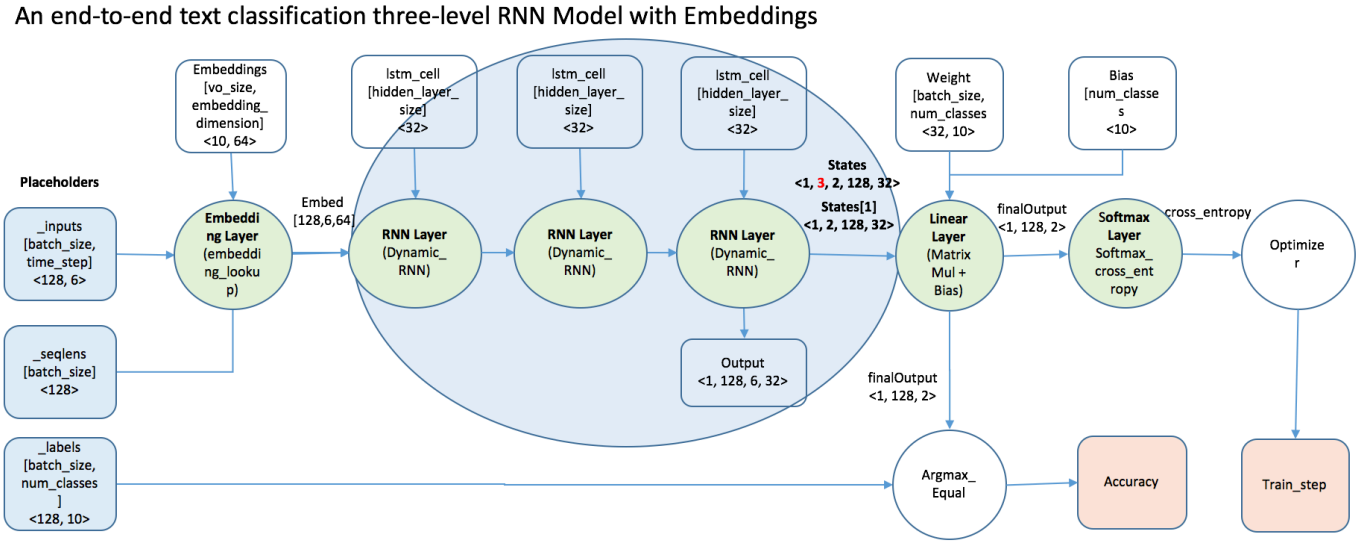
<Figure2>

An end-to-end text classification two-level RNN Model with Embeddings



The <Figure3> shows a change of two-level to three-level RNN. The shape of the states tensor is changed from <1, 2, 2, 128, 32> to <1, 3, 2 128, 32>. Also, the rest of them is all the same as the two-level RNN.

<Figure3>



Due to the no variation of the input sentence, which has only vocabulary from one to nine and 'pad' as 0, the training and test accuracy is 100% shown in <Figure4>

<Figure 4>

```
Accuracy at 0: 66.40625
Accuracy at 100: 100.00000
Accuracy at 200: 100.00000
Test batch accuracy 0: 100.00000
Test batch accuracy 1: 100.00000
Test batch accuracy 2: 100.00000
Test batch accuracy 3: 100.00000
Test batch accuracy 4: 100.00000
```

4. References

[1] Nikhil Buduma (2017). Fundamentals of deep learning. Sebastopol, CA: O'Reilly Media