Report for CS395T Project3: Neural Networks for Sentiment Analysis

Zhao Liu
Department of Mathematics
The University of Texas at Austin
zliu@math.utexas.edu

Abstract

In this project, our task is sentiment analysis by Neural Network Model. We first implement a Feed-Foward Neural Network in the style of Iyyer et al. (2015), then based on this foundation, we build a Recurrent Neural Network and complete the same sentiment task. We tried both undirectional LSTM and Bidirectional LSTM and compared their performance.

1 Introduction

In this project, we aim to classify sentences into two classes: positive or negative. Neural Network Models can be built to take into account both sentence content and word orders. They have the capability of capturing the underlying nonlinear, complex structures of the problem, and can be easily trained and tuned.

The first model we implement for sentiment analysis is the Deep Averaging Network proposed by Iyyer et al. (2015). Not taking into account the word orders, this model treat each input sentence as a bag of word embeddings. Namely, the key point is to take vector average of the word embeddings associated with a sentence, and feed it into a 2-layered FeedForward Neural Network (FFNN). Its structure is clearly shown in Figure 1 from the paper.

The second model is Recurrent Neural Network (RNN). With the same word embeddings, instead of taking unordered average, we treat each sentence as a sequence of word vectors and feed into an RNN model. To capture long-term dependencies, we apply the Long Short-Term Memory (LSTM) cells in the hidden layers for RNN, which is a solution to the vanishing gradient problem. We tried both undirectional and bidirectional LSTMs. The latter will run the input word sequence in two opposite directions, which helps preserve information from both the past and the future. The basic structure is shown in Figure **??**.

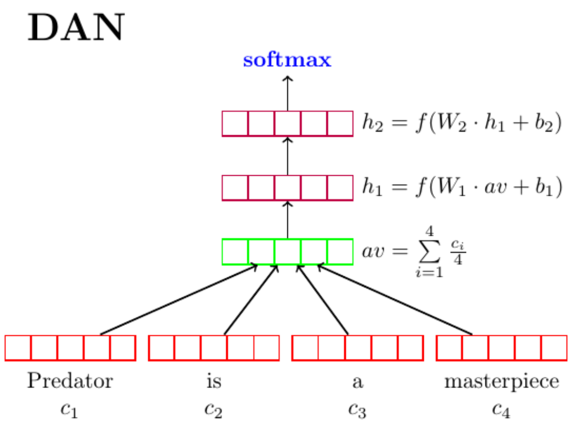

DAN
softmax
$h_2 = f(W_2 \cdot h_1 + b_2)$
$h_1 = f(W_1 \cdot av + b_1)$
$av = \sum_{i=1}^{4} \frac{c_i}{4}$
Predator
c_1
is
c_2
a
c_3
masterpiece
c_4

Figure 1: The FeedForward Neural Network by averaging

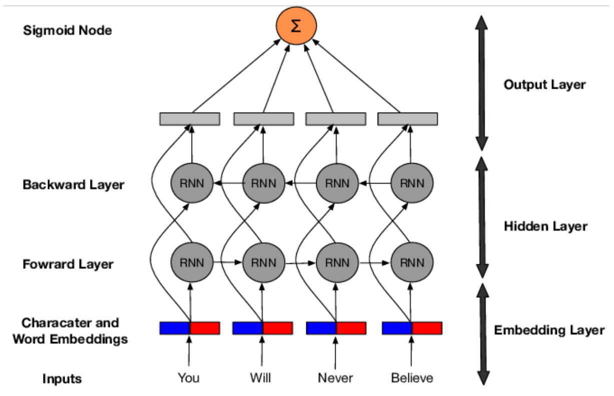


Figure 2: The structure of Bidirectional RNN

2 Implementation details

Using the `TensorFlow` library in Python is very convenient for Neural Networks: no need to worry about computing derivatives with back-propagation, or implementing the 'gates' of an LSTM block.

2.1 Building the Computation Graph

For FFNN, the main functions used in the program is shown in the flow chart Figure 3. The `labels`, `input_seq_lens`, `input_word_indicies` are the input tensors achieved by `tf.placeholder()`, the weight parameters `w_in_1` and `w_in_2` are variables, and `one_best` and `loss` are outputs. We just make several notices for the program:

- A small error in the provided code: in the `pad_to_length` function, which is applied to padding the word-indecies matrix to the maximum sentence length for taking average, instead of padding zeros, we should pad the index corresponds to "UNK" i.e. the index with all-zero word vector. The zero index actually corresponds to a nonzero vector and impacts taking averages afterwards.

- The batch size `mini_batch_size` can be tuned. Since we may have varying batch sizes in one program, the inputs (placeholders) have `None` in one of their dimensions. It is not convenient to explicitly include a bias 'b' as in Figure 1. So we pad the nodes `x`, `z` with 1's and assign larger dimensions for the weights to include the bias term.

- Choose hidden vector size = 100.

Figure 4 shows the main part of the Computation Graph of Bidirectional LSTM, which neglects the same parts in Figure 3. The inputs are simply word-indecies matrix and its revered ordered matrix, each sentence padded **in the front** by "UNK" index (implemented in `pre_pad_length` function). It might not be good to pad "UNK"s at the end of a sentence, which takes the most advantage of RNN. Also, note that `tf.unstack` is used to get a list of tensors to feed into RNN.

The structure for undirectional LSTM is similar to Figure 4 but with just one direction and one LSTM cell. For all LSTM cells, we use `forget_bias = 1.0`.

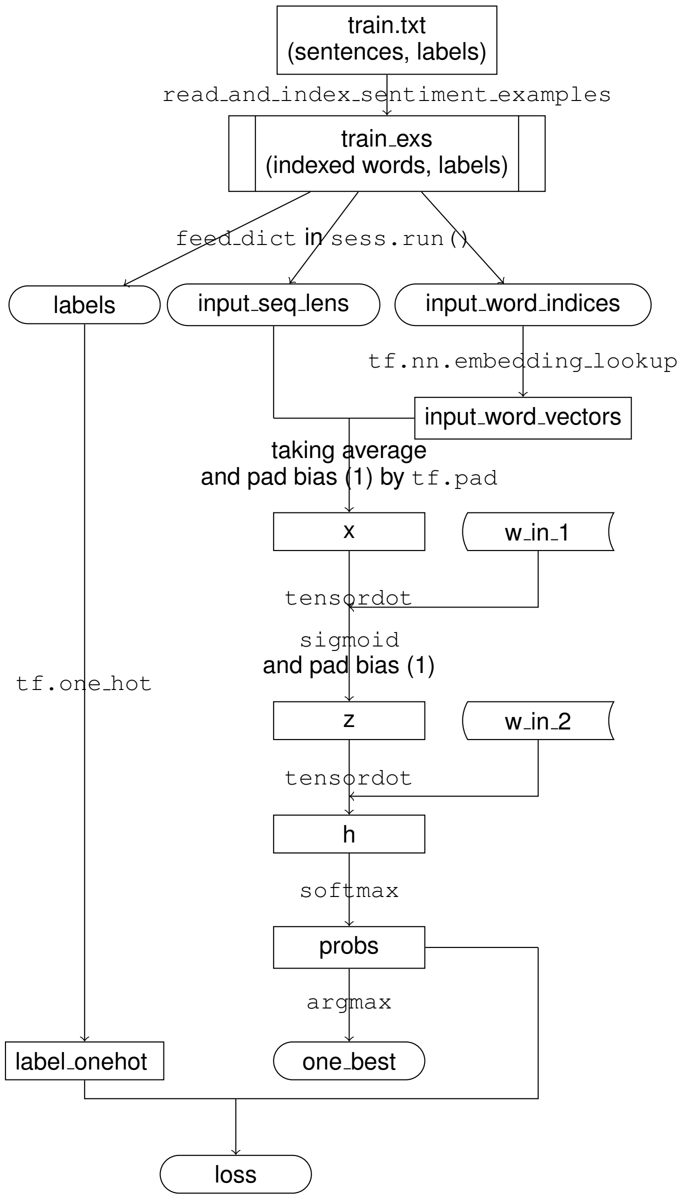


Figure 3: Computation Graph of FFNN implemented by TensorFlow

2.2 Training Algorithm Customization

- Initial_learning_rate = 0.005 for FFNN, and 0.001 for RNN.

- Learning_rate_decay_factor = 0.99, decay_steps = 1000

- Optimizer: Adam

- Glorot initializer for variables: (`tf.contrib.layers.xavier_initializer`).

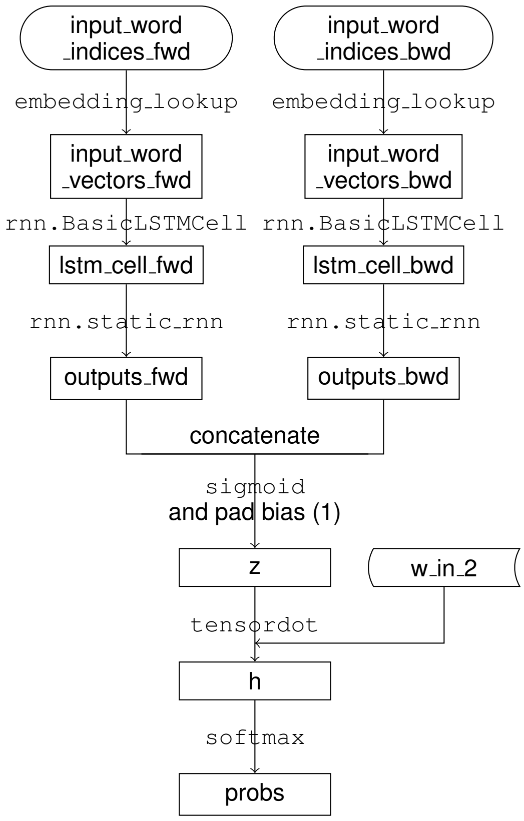


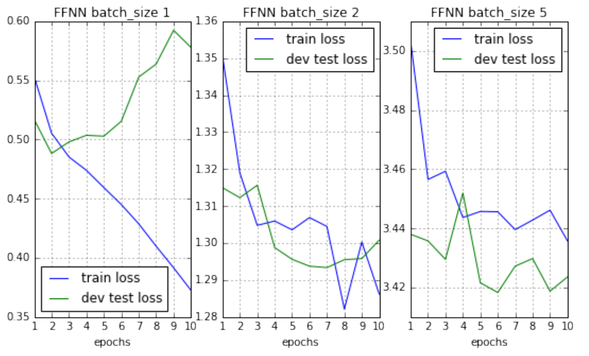
Figure 4: Computation Graph of Bidirectional LSTM implemented by TensorFlow

3 Results

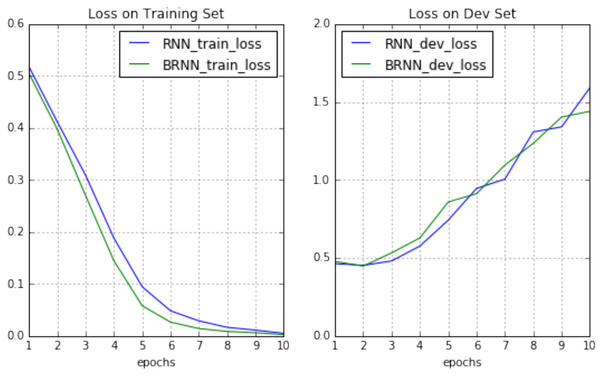
We first tried FFNN with different batch sizes, and compare their performance for `num_epochs = 10` runs. From Figure 5a, we observe that both their loss on training set decreases, but the loss on Dev test set increases after several steps, which suggests that the model is overfitting. In our test, the accuracy for `batch_size = 1` is the best: it stabilizes to about 75.5%. The larger batch sizes, the larger oscillations we have for the accuracies (and it won't work better even if we try `num_epochs = 50`), as shown in Figure 5c. The disadvantage of using larger batches might result from the small size of our dataset.

For RNN models the performances are significantly better. To our surprise, Undirectional LSTM can reach higher accuracy (80.5%) than Bidirectional LSTM (78.3%). Again, based on the graph on Training Loss and Test Loss (Figure 5b), the model reaches best performance within 2-3 epochs, and becomes overfitting in the following iterations.

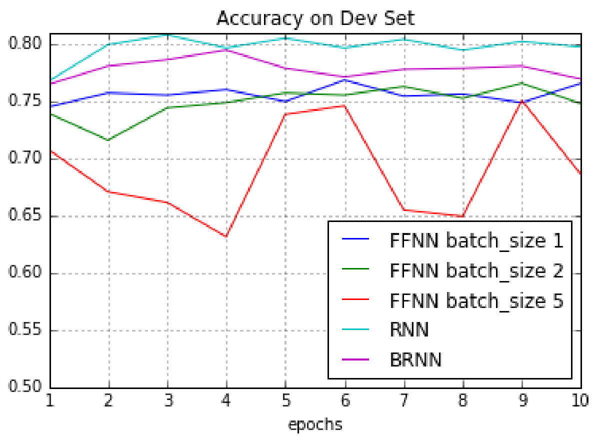
The submitted result is Undirectional LSTM with `num_epochs = 3`.



(a) Loss for FFNN with different batch sizes



(b) Loss for RNN



(c) Comparision of accuracies

References

Iyyer, Mohit, et al. 2015. *Deep unordered composition rivals syntactic methods for text classification.* Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, Vol.1.

Goldberg, Yoav. 2015. *A Primer on Neural Network Models for Natural Language Processing (Draft as of October 5, 2015).* http://u.cs.biu.ac.il/ yogo/nnlp.pdf