CS388: Natural Language Processing Mini 2

Nidhi Kadkol
University of Texas at Austin UTEID: nk9368
nidhikadkol@gmail.com

1 Problem Statement

In this mini project we look at a sentence from a move review database and classify it as a positive or negative review - whether the critic who wrote it is praising the movie or not. This is a problem of sentiment analysis, and we solve it by using a feed forward neural network and then a Recurrent Neural Network (RNN) using Long Short-Term Memory (LSTM) units. Collaborators - Shivangi Mahto, Ryan Westerman, Hailey Hultquist.

2 Implementation and Results

This is a classification problem into 1 of 2 classes - positive sentiment or negative sentiment. In the first part, I implemented a feed-forward neural network with tanh non-linearity and softmax classification. I initialized the weights with uniform xavier initialization. At first I used 1 hidden layer of dimension 20 and the accuracy on the development set was about **65%**. I then added a second hidden layer of 40 units and the accuracy on the development set increased drastically to **76.45%**. I trained this model for 15 epochs and used the Adam optimizer with an initial learning rate of 0.001. Both these values are using the 300 dimensional vector embeddings for the words. With the 50 dimensional vector and 2 hidden layers, we get a development set accuracy of **70.73%**.

For part 2, I chose to implement a unidirectional single layer LSTM with hidden size 100. I decided to use the pretrained 300-dimensional word embeddings and keep them as fixed, since they were working well for part 1 and fine-tuning them would increasing training time. Also, fine-tuning them further may result in overfitting. I used the Adam Optimizer and an initial learning rate of 0.001. I initialized the hidden layers to 0 after every iteration. I used a batch size of 1 and trained the model for 20 epochs initially. This model produced extremely good results right at the beginning, getting **80.58%** at the end of the third epoch. This could be primarily due to the fact that with a batch size of 1, gradient updates are happening much more frequently, and so within a few epochs the model performance peaks. I used the same hand-coded cross-entropy loss from the example code for both part 1 and part 2 to measure the model performance. For predicting the sentiment, I used the output from the last LSTM unit which is a 1D vector of size equal to hidden size, since the number of layers and batch size is 1. I pass this to a fully connected layer of size 2, which is the number of classes, and then softmax these values before I calculate the loss. The dev accuracy fluctuates between 78 and 79 for the next few epochs after epoch 3, then becomes **80.39%** in the 9th epoch. It starts overfitting very evidently after the 16th epoch as seen in figure 1. Based on these results, I decided to train my final model for 3 epochs. I also experimented with a bi-directional LSTM, with the remaining architecture and hyperparameters the same as above. This did not do significantly better, and gave a best development accuracy of **80.21%** on epoch 16.

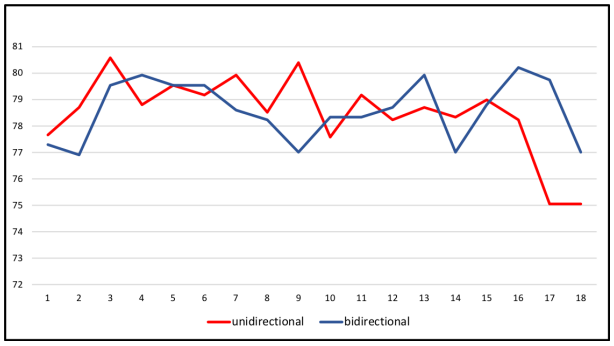

Figure 1: Development Accuracies for Unidirectional and Bidirectional LSTMs