NLP: Assignment 2 Report

Saraj Manes and Norbert Eke
Carleton S#: 300098949 and Carleton S#: 101102981

Task 1:

For task 1, the accuracies of the classification on the test set can be seen in Table 1.

In this task we have followed Steven Hewitt's "Textual entailment with TensorFlow" code tutorial, which was provided as a started code in the assignment. We decided to keep Standford's GloVe embeddings, as it looked like it is working well with the SNLI data set, but decided to try more than just the 50 dimensional models.

We adapted the code and its helper methods to fit the SICK data set's format, but kept the model architecture the same as in the starter code, as it was already proven to work fine for the same task on a different, but similar dataset (SNLI). We trained on SICK_train, and tested on SICK_test, while experimenting with various hyper parameters using the SICK_dev dataset.

Shortly summarizing our method firstly, the GloVe model is responsible to turn the two input sentences a sequence of word vectors, and then a helper method makes a sequence of vectors for each sentence. We followed the code tutorial and we are using a bi-directional RNN, with two different LSTM units, running both forward and backward through the input data. Dropout is used on the first and last layers from the unrolled LSTM network portions. The output from the LSTMs is passed into a fully connected layer, then a software activation function is used on the final output layer to output the probability of the sentences being entailment, neutral or contradiction. The loss function used in this model is categorical cross entropy.

Experiment Set-Up:

- We have experimented with *learning rate 0.01 and 0.001*.
- The *number of epochs* stayed constant at *25*. We believe that by that iteration the results have stabilized enough.
- We have used *batch size 128* when experimenting with *64 hidden layers*, while using *batch size 256* when constructing a neural network with *128 hidden layers*.
- In terms of word embeddings we have used the GloVe.6B pre-trained embeddings, and we experimented with 2 different sized models: 300 and 50 dimensions.

Model Number	Learning rate	Hidden Layers	Batch Size	Epochs	GloVe.6B Dim	Accuracy
#1	0.01	64	128	25	300	0.55652
#2	0.001	64	128	25	300	0.56687
#3	0.01	128	256	25	300	0.56687
#4	0.001	128	256	25	300	0.56687
#5	0.01	64	128	25	50	0.55206
#6	0.001	64	128	25	50	0.56687
#7	0.01	128	256	25	50	**0.60402**
#8	0.001	128	256	25	50	0.56687

Table 1: All experiments ran Task 1 - Text Entailment. Highest accuracy is in red.

The *best model* for *Task 1* is **Model #7,** achieving accuracy just above **60%.** This model was trained 25 epochs, used 0.01 as a learning rate, had 128 hidden layers, used 256 as batch size and it achieved this performance using the GloVe.6B.50d word embedding model. Just as a note that most of the models overfit on the data, and classify a new prediction as the most common class. Unfortunately we could not develop different model architectures, thus we had to settle for really good hyper-parameters that outperform the baseline 56% accuracy result.

Task 2:

For task 2, the Pearson's Correlation Coefficient on the test set can be seen in Table 2.

In this task we used similar overall model architecture as in Task 1, except that we transformed the model from a classification task to a regression task. This means taking off the soft-max activation function from the output layer, changing the output layer from size 3 (probability of three different classes to be predicted) to size 1 (one value to be predicted) and changing the loss function from categorical cross entropy to mean squared error loss function. The rest of the model architecture and overall ML algorithm is the same, while we experimented with different hyper-parameters to find the best performing model.

Experiment Set-Up for Task 2 is the same as Task 1, except that we are measuring Pearson's Correlation Coefficient, not accuracy.

Model Number	Learning rate	Hidden Layers	Batch Size	Epochs	GloVe.6B Dim	Pearson's Correlation Coefficient
#1	0.01	64	128	25	300	0.53771
#2	0.001	64	128	25	300	0.03696
#3	0.01	128	256	25	300	0.48091
#4	0.001	128	256	25	300	0.02389
#5	0.01	64	128	25	50	0.57617
#6	0.001	64	128	25	50	0.04183
#7	0.01	128	256	25	50	**0.58408**
#8	0.001	128	256	25	50	0.07591

Table 2: All experiments ran Task 2 – Semantic Relatedness. Highest accuracy is in red.

The *best model* for *Task 2* is **Model #7**, achieving a Pearson Correlation Coefficient of **0.58408**. This model was trained 25 epochs, used 0.01 as a learning rate, had 128 hidden layers, used 256 as batch size and it achieved this performance using the GloVe.6B.50d word embedding model.

Just as a note, we noticed that for this task a learning rate of 0.001 does not work well, as the Pearson Correlation Coefficient is very low. Interestingly such learning rate still works for Task 1

Work sharing:

The work was divided equally between me and my partner, as we both completed 1 on the questions.

Running instructions:

Run TextEntailment.py for Task 1 and SemanticRelatedness.py for Task 2.

The embedding model files are not in the submission, as they would be too large to submit online, but they are easily downloadable. The test and training data is present in the /data/ directory, while the output is generated to the /output/ directory.

Library dependencies include: tensorflow, numpy, scipy, tqdm, sklearn and csv.