NLP Project Report
Daniel Crawford

Task
Predict whether or not a sentence will be entailed by its premise.

Model Structure
My model is constructed in several layers, where the number of LSTM layers is a hyperparameter. All inputs first go in through the embedding layer, then a bi-directional LSTM layer, then all the LSTM layers included, then they are summed up by column and put through a linear function to produce the output.

Important Notes
- For all models seen below, a train/test split of 80%/20% was used. This was used to combat the lack of data available already.
- After the model finishes all of its epochs, the best model is saved based on the smallest validation loss available. This is to avoid overfitting/overtraining with too many epochs.
- The optimizer used is the standard gradient classifier.
- The criterion used is binary cross entropy.
- All throughput is based on the performance of the GPU, not the CPU.
- Unless specified, all tests are ran with a batch size of 1.

Results
Our results show that this model is very difficult to fit without much more data. However, there have been some decent results with some good parameter tuning.
For reference, in our case our max length is the same as our embedding dimension.

- Throughput: This will change dramatically based on the hidden dimensions and max length of the list. Here are a few examples
    - Max length = 10, Hidden dimension = 60, Number of extra LSTM layers = 2
        - Throughput = 5.36 seconds
    - Max length = 30, Hidden dimension = 50, Number of extra LSTM layers = 2
        - Throughput = 7.52 seconds
    - Max length = 100, Hidden dimension = 50, Number of extra LSTM layers = 2
        - Throughput = 20.02 seconds
    - These appear to be mostly affected by our max length, so there is a massive tradeoff for features here.
- Performance: Because of our small dataset, it was smart to use smaller values for max length and hidden dimensions. Otherwise, there would be too many parameters to train.
    - Adjusting number of epochs:
        - Max length = 10, Hidden dimension = 60, Number of extra LSTM layers = 2, Number of epochs = 20

        - Accuracy: 56.9%
        - Precision: 54.2%
        - Recall: 92.7%
        - F-Score: 68.4%
    - Max length = 10, Hidden dimension = 60, Number of extra LSTM layers = 2, Number of epochs = 50
        - Accuracy: 51.1%
        - Precision: 51.2%
        - Recall: 59.4%
        - F-score: 55.03%
    - Looking at these two results, we can see that while the number of epochs can be useful, there is also the need to luck in randomization at the start.
- More features, less parameters to train:
    - Max length = 30, Hidden dimension = 50, Number of extra LSTM layers = 0, Number of epochs = 50
        - Accuracy: 52.5%
        - Precision: 52.9%
        - Recall: 52.1%
        - F-Score: 52.6%
- Using significantly more LSTM layers:
    - Max length = 20, Hidden dimension = 20, Number of extra LSTM layers = 5, Number of epochs = 20. **Batch size is 10 this time.**
        - Accuracy: 0.5036
        - Precision: 0.5036
        - Recall: 1.0
        - F-Score: 0.669
    - In this case, our model struggled to converge, and thus was never able to make good guesses so it only output one prediction.

Based on our results from these tests, we can see that **simpler models are better** when we're dealing with a very limited training set. We saw that giving significantly more parameters to train made it much more difficult for the model to learn. On the other hand, when we gave more features we saw that it was able to improve to a point, but likely it would struggle as more features would require more parameters to train. However, when we used a reasonable model with not too many parameters, were capable of making a model with decent predictive power.