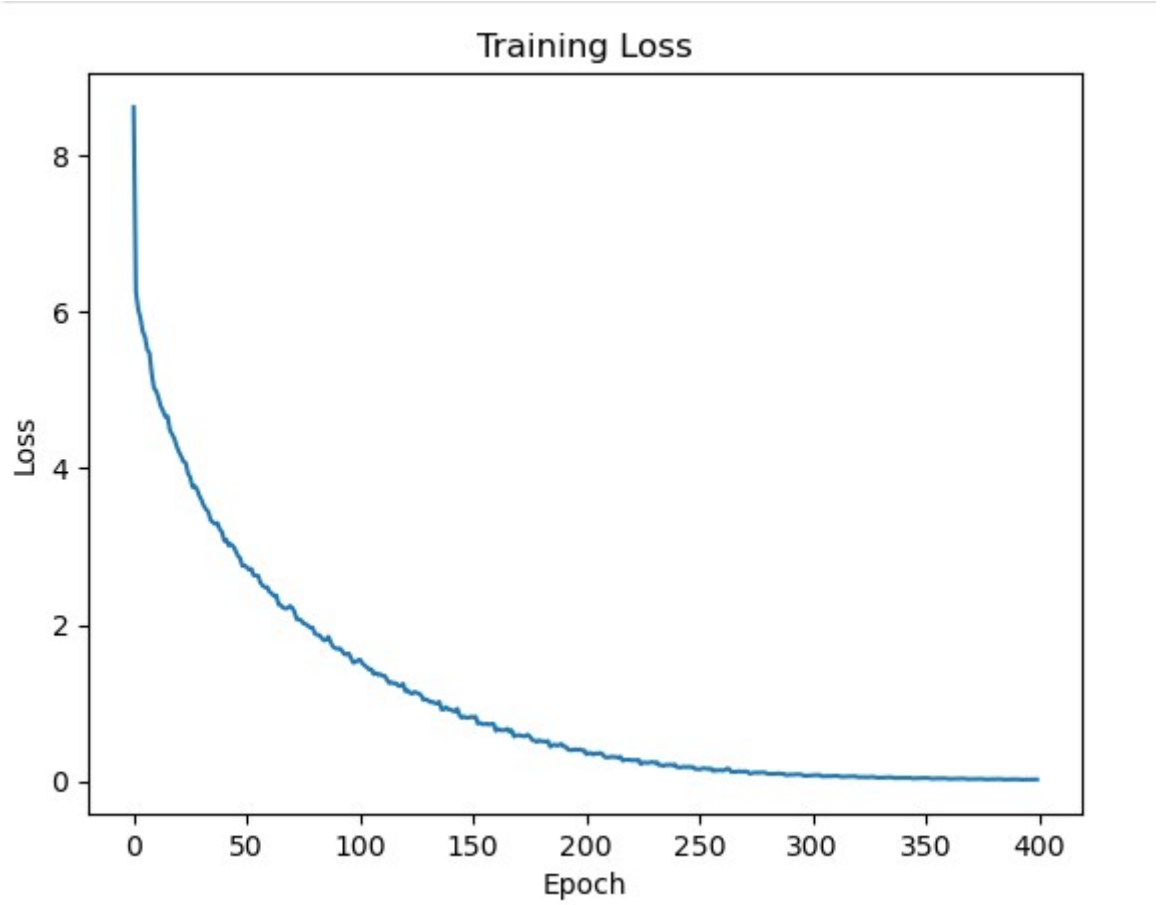
CS7.401: Introduction to NLP — Assignment 4

Roll no: 2022201024

1. Sentiment Analysis

Elmo-Model Training Loss v/s Epoch Graph




```
Final Training Loss: 0.0215792813396547
```

Loss v/s Epoch Graph for Sentiment Analysis Model

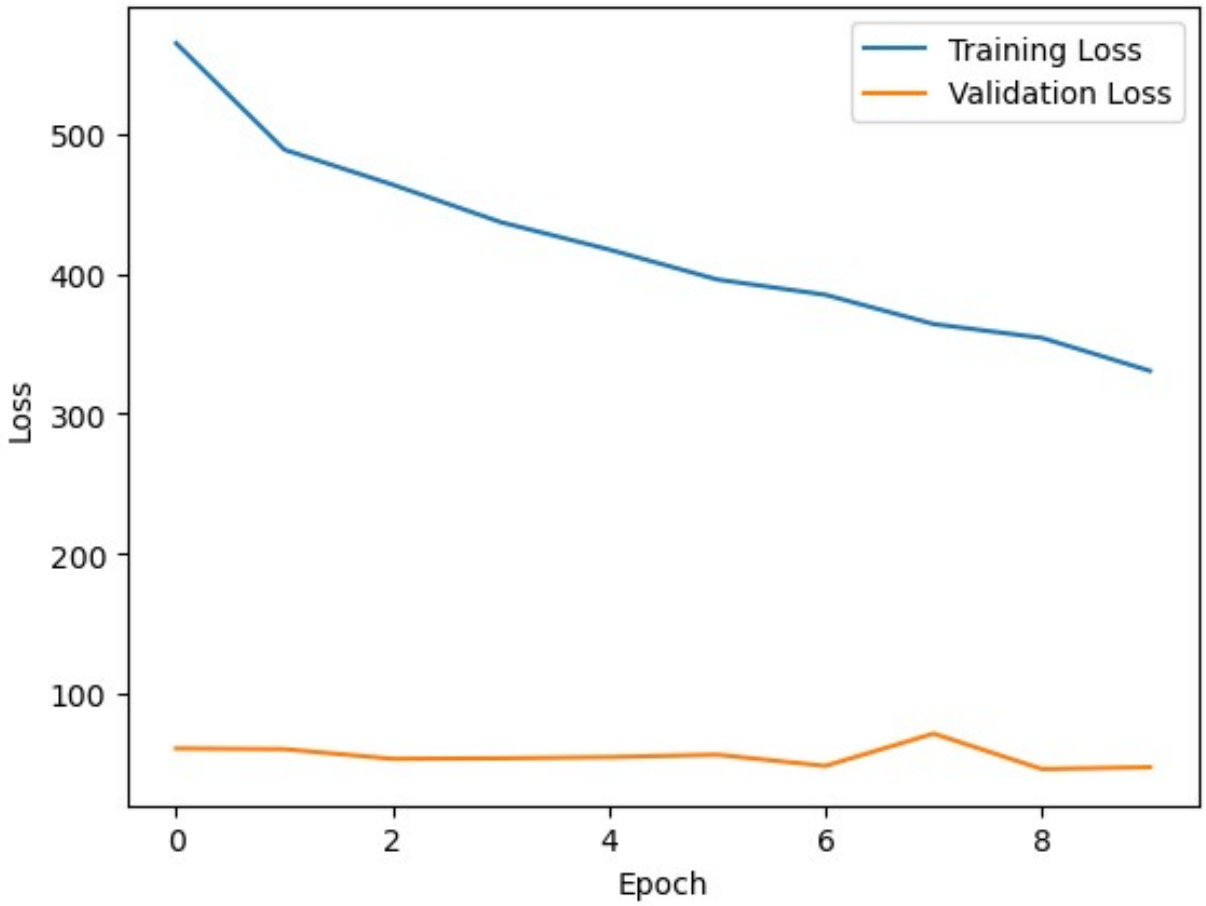
Epoch 1/10, Train Loss: 0.0661, Val Loss: 0.0552
Epoch 2/10, Train Loss: 0.0572, Val Loss: 0.0548
Epoch 3/10, Train Loss: 0.0543, Val Loss: 0.0485
Epoch 4/10, Train Loss: 0.0512, Val Loss: 0.0488
Epoch 5/10, Train Loss: 0.0489, Val Loss: 0.0496
Epoch 6/10, Train Loss: 0.0464, Val Loss: 0.0512
Epoch 7/10, Train Loss: 0.0451, Val Loss: 0.0439
Epoch 8/10, Train Loss: 0.0426, Val Loss: 0.0649
Epoch 9/10, Train Loss: 0.0415, Val Loss: 0.0418
Epoch 10/10, Train Loss: 0.0387, Val Loss: 0.0431

This training data shows the training and validation loss for a sentiment analysis task using an LSTM model. The model is defined in the code as having an input size, hidden size, number of layers, and output size, as well as a dropout rate for regularization. The LSTM layer is bidirectional, meaning it processes the input sequence in both forward and backward directions. The output of the

LSTM layer is then passed through three fully connected layers, with the final output being a single scalar value representing the sentiment score.

Looking at the training and validation loss over the ten epochs, we can see that the model's performance improves significantly in the first few epochs, with the training loss dropping from 0.0661 to 0.0489. The validation loss also drops from 0.0552 to 0.0496 during this period. However, after epoch 4, the model's performance begins to level off, and the validation loss even increases in epoch 8 before dropping again in epochs 9 and 10.

Overall, the model seems to be performing reasonably well, with both the training and validation loss reaching values below 0.05. However, there is some indication that the model may be starting to overfit to the training data in the later epochs, as the validation loss begins to increase again.



```
lstm_model.eval()
mse = 0.0
with torch.no_grad():
    for data, labels in sst_test_loader:
        lstm_input = data.float().permute(1, 0, 2).to(device)  # reshape to (seq_len, batch, input_size) and move to GPU
        outputs = lstm_model(lstm_input)
        labels = labels.float().to(device) # move labels to GPU
        mse += ((outputs - labels) ** 2).sum().item()  # add up the squared errors

mse /= len(sst_test_dataset)  # divide by the number of samples to get the average MSE
print(f'Test MSE: {mse:.4f}')
```

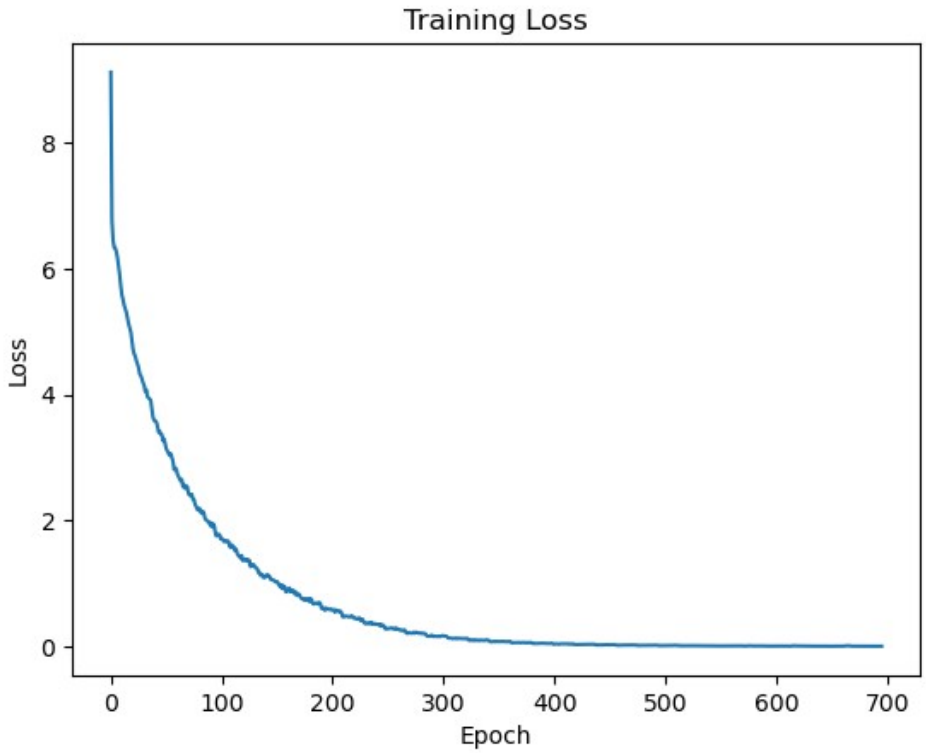
```
Test MSE: 0.0454
```

Test MSE = 0.0454

A test MSE of 0.0454 suggests that, on average, the squared difference between the predicted values and the actual values is 0.0454. This value is relatively low, indicating that the model has achieved good performance on the test set.

2. Natural Language Inference

Elmo-Model Training Loss v/s Epoch Graph



Loss v/s Epoch Graph for Sentiment Analysis Model

Epoch 1/10, Train Loss: 0.0169, Val Loss: 0.0165
Epoch 2/10, Train Loss: 0.0163, Val Loss: 0.0163
Epoch 3/10, Train Loss: 0.0160, Val Loss: 0.0163
Epoch 4/10, Train Loss: 0.0159, Val Loss: 0.0159
Epoch 5/10, Train Loss: 0.0156, Val Loss: 0.0161
Epoch 6/10, Train Loss: 0.0153, Val Loss: 0.0161
Epoch 7/10, Train Loss: 0.0150, Val Loss: 0.0164
Epoch 8/10, Train Loss: 0.0145, Val Loss: 0.0166
Epoch 9/10, Train Loss: 0.0136, Val Loss: 0.0170
Epoch 10/10, Train Loss: 0.0125, Val Loss: 0.0180

The training data shows the performance of a natural language inference task using an LSTM-based model with attention. The model has three layers, including an LSTM layer, an attention layer, and a fully connected layer. The model takes two inputs (premise and hypothesis embeddings) and outputs the probability distribution over the three possible classes (entailment, contradiction, and neutral).

The training process consists of ten epochs, during which the model is trained using the Adam optimizer with a learning rate of 0.001. The training and validation losses are calculated at the end of each epoch. The training loss decreases gradually over the epochs from 0.0169 to 0.0125, while

the validation loss fluctuates but remains around 0.016. This indicates that the model is performing well on both the training and validation sets and is not overfitting.

In conclusion, the provided training data shows the successful training of a natural language inference model using an LSTM-based architecture with attention. The model achieved good results with low training and validation losses, indicating that it has learned to effectively classify the input sequences into one of the three classes.