Assignment 3: Sentiment Classification

Learning Curves

1. Random Embeddings

Sentiment Classification Accuracy of 52.20%

Confusion matrix:

[[518. 393.]

[477. 432.]]

2. GloVe Embedings

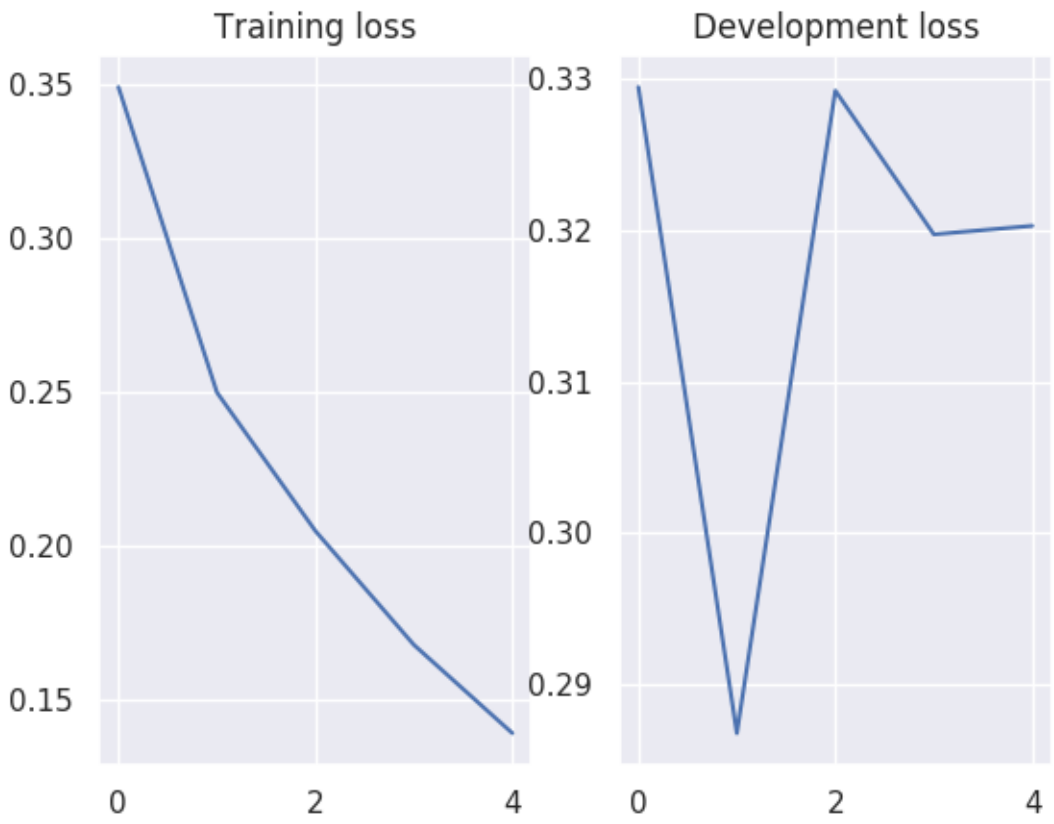
Sentiment Classification Accuracy of 84.12%

Confusion matrix:

[[731. 180.]

[109. 800.]]

3. ELMO Embeddings



Sentiment Classification Accuracy of 89.01%

Confusion matrix:

[[782. 129.]

[71. 838.]]

4. ELMO + GloVe Embeddings

Sentiment Classification Accuracy of 89.62%

Confusion matrix:

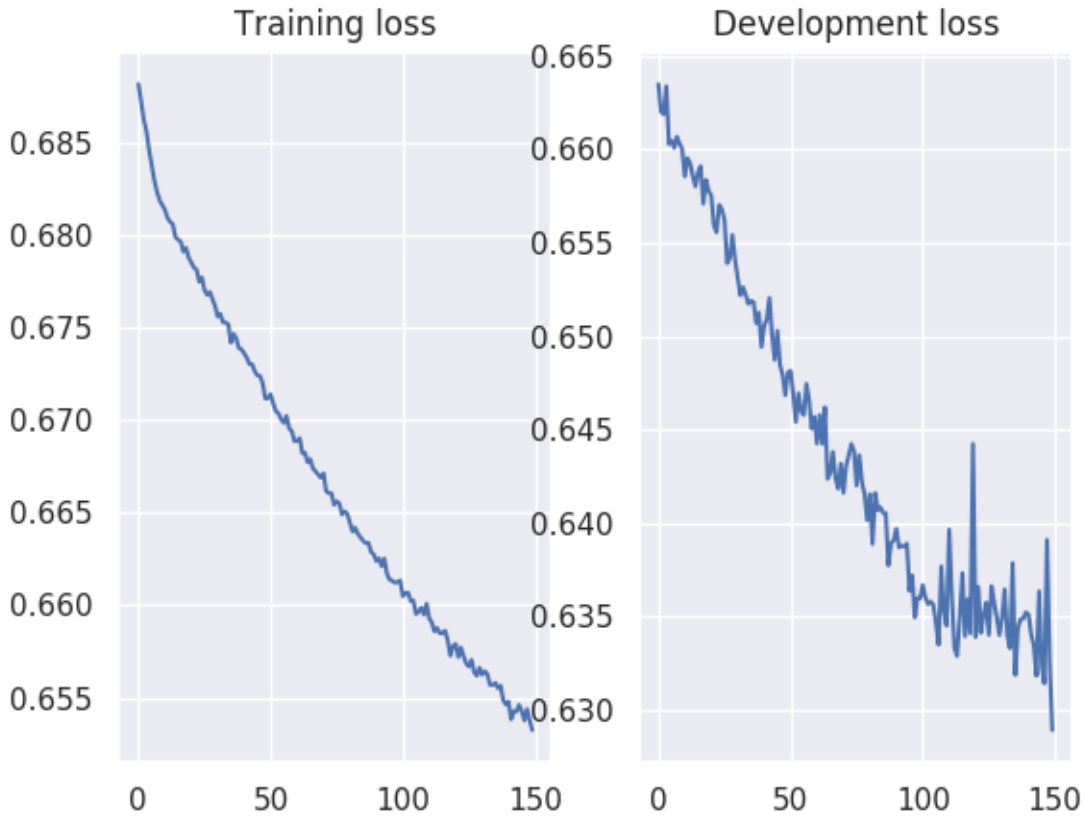
tensor([[791., 120.],

[69., 840.]])

Hyperparameter Experiments

Experiment 1: Does training the Sentiment Network with random embeddings over more epochs rival ELMO/Glove Embeddings?

Motivation: Training with random embeddings is a lot faster than training with Elmo or even Glove. If we look at the training loss curve from training with random embeddings, it has not plateaued by 10 epochs. I suspect that if we just train with random embeddings for more epochs, its performance could be more comparable to Elmo or Glove.



Results: As you can see above, I trained with random embeddings for 150 episodes. Training time was similar to Glove embeddings and lower than Elmo embeddings.

Sentiment Classification Accuracy of 59.12%

Confusion matrix:

tensor([[538., 373.],

[371., 538.]])

The classification accuracy went up from 52% (10 epochs) to 59% (150 epochs). However, the classification accuracy didn't come close to the >80% accuracy when using Glove or Elmo.

Experiment 2: Impact of increasing the RNN hidden size on Glove vs Elmo

The results in the previous section were obtained with the given hyperparameter setting of the RNN hidden size of 1024. I changed that hidden size to 2048 and observed the impact on validation accuracy for glove and elmo embeddings.

Results:

Glove Embeddings with 2046 hidden size:

Sentiment Classification Accuracy of 82.14%

Confusion matrix:

tensor([[702., 209.],

[116., 793.]])

Elmo Embeddings with 2048 embedding size:

Sentiment Classification Accuracy of 90.27%

Confusion matrix:

tensor([[828., 83.],

[94., 815.]])

Thus we can see that increasing the layer size had a big impact on sentiment classification with ELMO.

Overfitting

For all embedding methods, my loss curve on the validation data was increasing with the number of epochs – suggesting severe overfitting of the model to the training data. To mitigate overfitting, I introduced dropout (with probability of dropping a neuron set to 0.5) after the non-linear activations of the classification module in SentimentNetwork.

I reduced the capacity of my classification module by using just one linear layer with 2 intermediate nodes. While this resulted in the dev loss to fall with more epochs, it resulted in my classifier always predicting the same class. As a result, I decided that I was going to have to overfit to the training data in order to get a better classification accuracy.

I also found that if I trained for more epochs (for random embeddings), I was no longer overfitting and the dev loss started to go down.