Allison Aprile
Assignment 4
April 30, 2021

Sentiment Analysis

Preprocessing Steps

For both the positive and negative review files, I first loaded the text as a string. Because it is structured in an HTML format, I instantiated a BeautifulSoup reader object on the text. Using the BeautifulSoup find_all function, I was able to filter the text into a list of strings between the 'review_text' tags. I then removed any newline and punctation characters, as well as normalized the text by lowercasing it. Finally, I tokenized the text. Each sequence was also padded to a length of 75, numerically encoded based on the vocabulary, and the corresponding label vectors were one-hot encoded.

I split the data into 80% training (1,600 samples), 10% validation (200 samples), and 10% testing (200). The testing set is balanced, with 100 samples belonging to each the positive and negative classes.

I initialized the Embedding layer using the GLoVe 50d pretrained embeddings. These were set to trainable during training.

CNN

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_60 (Embedding)     (None, 75, 50)            951700
_________________________________________________________________
conv1d_38 (Conv1D)           (None, 75, 64)            9664
_________________________________________________________________
max_pooling1d_49 (MaxPooling (None, 37, 64)            0
_________________________________________________________________
flatten_47 (Flatten)         (None, 2368)              0
_________________________________________________________________
dense_93 (Dense)             (None, 1)                 2369
=================================================================
Total params: 963,733
Trainable params: 963,733
Non-trainable params: 0
_________________________________________________________________
```

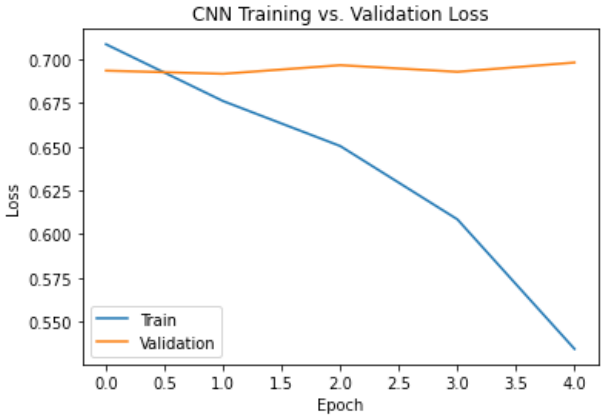
Hyperparameters: Number of filters = 64, kernel size = 3x3

For the CNN, I tried 16, 32, 64, and 128 filters, as well as 1x1 and 3x3 for the kernel size. I also tried no Dense layer and Dense layers with 10 and 100 neurons for the top layer MLP. Surprisingly to me, the lower numbers of filters performed better, observing the best validation loss/accuracy at 64.

Allison Aprile
Assignment 4
April 30, 2021

However, for all combinations, the performance margin was very small. Because CNN is SOTA (moreso for computer vision, but upcoming for NLP tasks), I was disappointed in this performance. I tried increasing/decreasing the input sizes of the word vectors (25, 50, 75, 100) and unfreezing/freezing the Embedding layer, but all efforts did not have an impact on performance.

Loss function: Binary cross-entropy
Learning rate: 1e-3
Optimizer: Adam



Results:
Testing accuracy: 0.53

LSTM (All Hidden States)

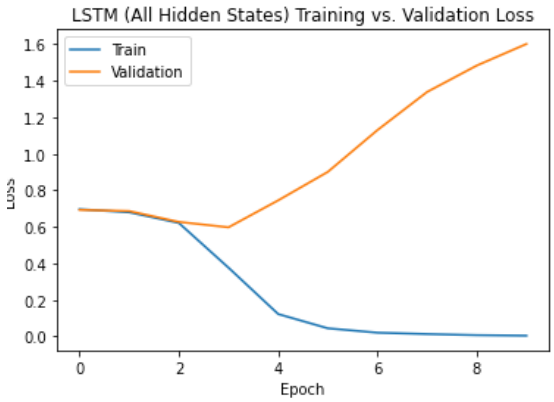
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_61 (Embedding)     (None, 75, 50)            951700
_________________________________________________________________
bidirectional_23 (Bidirectio (None, 75, 32)            8576
_________________________________________________________________
max_pooling1d_50 (MaxPooling (None, 37, 32)            0
_________________________________________________________________
flatten_48 (Flatten)         (None, 1184)              0
_________________________________________________________________
dense_94 (Dense)             (None, 1)                 1185
=================================================================
Total params: 961,461
Trainable params: 961,461
Non-trainable params: 0
_________________________________________________________________
```

Allison Aprile
Assignment 4
April 30, 2021

Hyperparameters: Hidden state size = 16

For this LSTM, I had a similar experience as with the CNN - when trying the varying hidden state sizes (16, 32, 64, 128), I noticed that the lower size led to better performance. I also observed that taking the element-wise maximum (verus average) of all the states after the Bidirectional LSTM layer yielded better performance. Ultimately, this is not ideal performance (only slightly better than random guess), but it still learned more than the CNN.

Loss function: Binary cross-entropy
Learning rate: 1e-3
Optimizer: Adam



Results:
Testing accuracy: 0.63

LSTM (Final Hidden State)

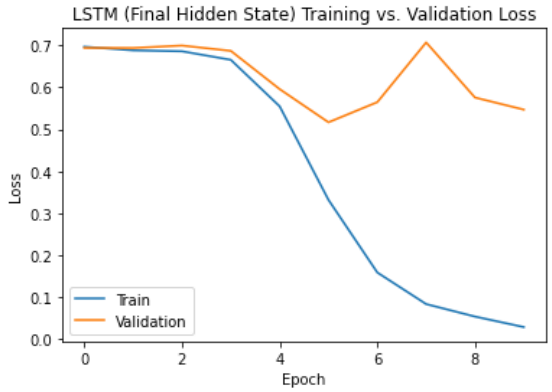
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_62 (Embedding)     (None, 75, 50)            951700
_________________________________________________________________
bidirectional_24 (Bidirectio (None, 32)                8576
_________________________________________________________________
dense_95 (Dense)             (None, 1)                 33
=================================================================
Total params: 960,309
Trainable params: 960,309
Non-trainable params: 0
_________________________________________________________________
```

Allison Aprile
Assignment 4
April 30, 2021

Hyperparameters: Hidden state size = 16

For this LSTM, I followed the same hyperparameter tuning as with the CNN and prior LSTM. I once again found that unfreezing the Embedding layer, using a hidden state size of 16 yieled the best validation performance. Surprisingly, it learned more than the LSTM that used all of the hidden states.

Loss function: Binary cross-entropy
Learning rate: 1e-3
Optimizer: Adam



Results:
Testing accuracy: 0.775

Allison Aprile
Assignment 4
April 30, 2021

Model Comparison

Based on the below table, it is clear that the LSTM model (using only the final hidden state) is a viable model. The CNN and LSTM (using the element-wise maximum of all the hidden states) are barely better than 'random guess', despite extensive hyperparameter tuning and experimentation with the padded sequence length. With further evaluation and processing of the dataset, possibly the models could be improved.

Model	Accuracy
CNN	0.530
LSTM (All Hidden States)	0.630
LSTM (Final Hidden State)	0.775