Assignment 4 Writeup
Ryan Do
MIE324
November 9, 2018

7. Experimental and Conceptual Questions

1. *(5 points) After training on the three models, report the loss and accuracy on the train/validation/test in a total. There should be a total of 18 numbers. Which model performed the best? Is there a significant difference between the validation and test accuracy? Provide a reason for your answer.*

	Train		**Validation**		**Test**	
	Accuracy	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**
Baseline	87.4%	0.401	87.7%	0.274	86.1%	0.251
RNN	99.6%	0.0115	91.3%	0.135	91.7%	0.114
CNN	100%	0.00474	91.1%	0.164	91.3%	0.166

The RNN model performed the best, edging out the CNN by a slight margin. This makes sense because of the LSTM allowing consideration of sequential history. There was not a significant difference between validation and test accuracy. In fact, test accuracy was higher in the RNN and CNN cases. The reason for this is that the test data came from the same distribution/ source as the validation data.

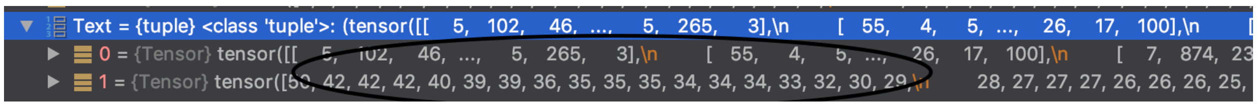
2. *(5 points) In the baseline model, what information contained in the original sentence is being ignored? How will the performance of the baseline model inform you about the importance of that information?*

In the baseline model, the vector embeddings of the individual words are being ignored. The baseline model simply takes the average of the word vectors in a sentence to produce a resultant vector that represents an overall arbitrary *meaning* of the sentence. The information captured by the presence of specific words is not retained. The baseline model performs significantly worse than the RNN and CNN implementations (~5-6% lower test accuracy), however it still performs quite well with a test accuracy of 86%. This tells me that capturing individual words is important enough to give a performance boost, but the overall subjective/objective characteristic of a sentence can be sufficiently captured with a vector average.

3. *(15 points) For the RNN architecture, examine the effect of using pack padded sequence to ensure that we did indeed get the correct last hidden state (Figure 4 (Right)). Train the RNN and report the loss and accuracy on the train/validation/test under these 3 scenarios:*
    a. *Default scenario, with using pack padded sequence and using the BucketIterator*
    b. *Without calling pack padded sequence, and using the BucketIterator*
    c. *Without calling pack padded sequence, and using the Iterator.*

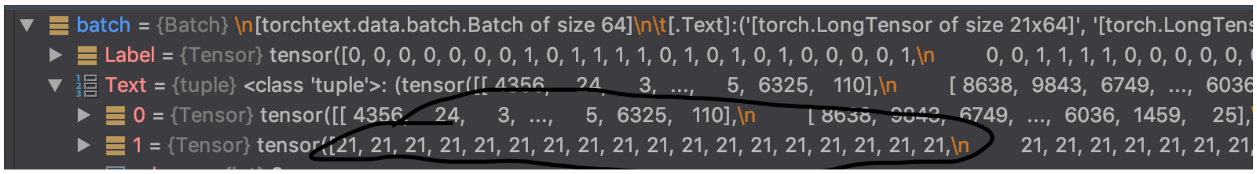
What do you notice about the lengths of the sentences in the batch when using Iterator class instead? Given the results of the experiment, explain how you think these two factors affect the performance and why.

	Train		**Validation**		**Test**	
	Accuracy	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**
w/ pack padded sequence and Bucket Iterator	99.9%	0.00379	91.63%	0.223	91.1%	0.335
w/o pack padded sequence and w/ BucketIterator	99.2%	0.0234	90.6%	0.542	91.6%	0.00204
w/o pack padded sequence and w/ Iterator	97.5%	0.0730	88.7%	0.260	87.9%	0.315



```
▼ Text = {tuple} <class 'tuple'>: (tensor([[    5,  102,   46, ....,    5,  265,    3],\n       [  55,    4,    5, ...,   26,   17,  100],\n       [
  ▶ 0 = {Tensor} tensor([[    5,  102,   46, ...,    5,  265,    3],\n       [  55,    4,    5, ...,   26,   17,  100],\n       [   7,  874,  23
  ▶ 1 = {Tensor} tensor([50, 42, 42, 42, 40, 39, 39, 36, 35, 35, 35, 34, 34, 34, 33, 32, 30, 29,\n        28, 27, 27, 27, 26, 26, 26, 25,
```

Lengths of sentences in a batch using Iterator class



```
▼ batch = {Batch} \n[torchtext.data.batch.Batch of size 64]\n\t[.Text]:('[torch.LongTensor of size 21x64]', '[torch.LongTens
  ▶ Label = {Tensor} tensor([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,\n        0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0,
  ▼ Text = {tuple} <class 'tuple'>: (tensor([[ 4356,   24,    3,  ...,    5,  6325,   110],\n       [ 8638,  9843,  6749,  ...,  6036
    ▶ 0 = {Tensor} tensor([[ 4356,   24,    3,  ...,    5,  6325,   110],\n       [ 8638,  9843,  6749,  ...,  6036,  1459,    25],
    ▶ 1 = {Tensor} tensor([21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21,\n        21, 21, 21, 21, 21, 21, 21,
```

Lengths of sentences in a batch using BucketIterator class.

When using the Iterator class over the BucketIterator class, each batch contains sentences with a range of lengths. On the other hand, the BucketIterator class batches sentences with similar lengths together.
The results of the experiment showed that pack padded sequence had minimal effect on the classification accuracy when using BucketIterator. However, when using Iterator, omitting pack padded sequence caused a drop in performance.
Here is how these factors affect the performance:
Since BucketIterator batches sentences with similar length, pack padded sequence has minimal effect because there generally very little to pad. Iterator batches sentences with different lengths, so pack padded sequence is required to have the RNN output the correct hidden state for each sentence. Omitting pack padded sequence using the Iterator therefore includes padded hidden states which would distort the final vector going into the fully connected layer. This leads to a less consistent classification thus a drop in performance.

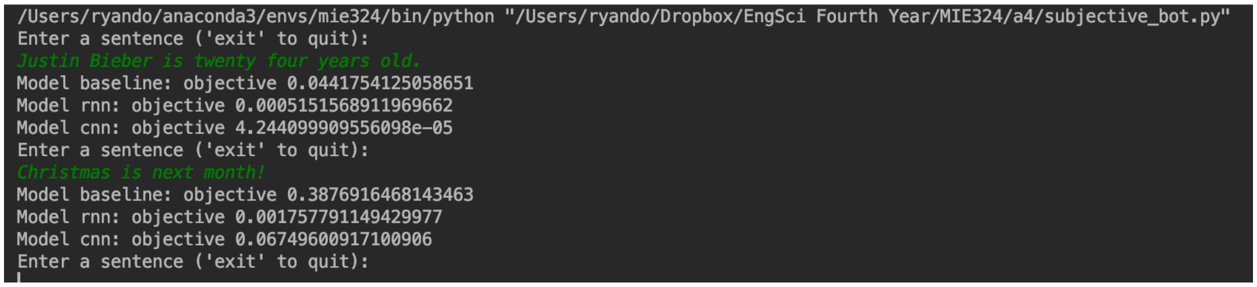
4. *(10 points) In the CNN architecture, what do you think the kernels are learning to detect? When performing max-pooling in the convolution activations, what kind of information is the model discarding? Compare how this is different or similar to the baseline model's discarding of information.*

In the CNN architecture, the kernels are learning to detect patterns that match semantic meaning to subjectivity/ objectivity, from sub-sequences of adjacent words. The kernels (of size 2 and 4) are *scanning* to pick-up indicators of subjectivity/ objectivity from 2 or 4 word sequences.

When performing max-pooling in the convolution activations, the model is only retaining a single element of a feature map that corresponds with the highest correlation between the kernel and data. It is essentially discarding information from words that are deemed irrelevant to objectivity/subjectivity. This is similar to the baseline model as they both take a single measure from an entire sentence to characterize subjectivity (average embedding in the case of the baseline and max of a feature map in the case of the CNN). However, this is different because simply taking an average of all the words does not retain any patterns in the semantics of a sentence whereas taking the maxpool reinforces specific patterns that correlate to subjectivity/objectivity.
For example, an average that corresponds to 100% subjectivity can be coincidentally produced by a very large number of combinations of random words.

5. *(10 points) Try running the subjective bot.py script on 4 sentences that you come up with yourself, where 2 are definitely objective/subjective, while 2 are borderline subjective/objective, according to your opinion. Include your console output in the write up. Comment on how the three models performed and whether they are behaving as you expected. Do they agree with each other? Does the majority vote of the models lead to correct answer for the 4 cases? Which model seems to be performing the best?*

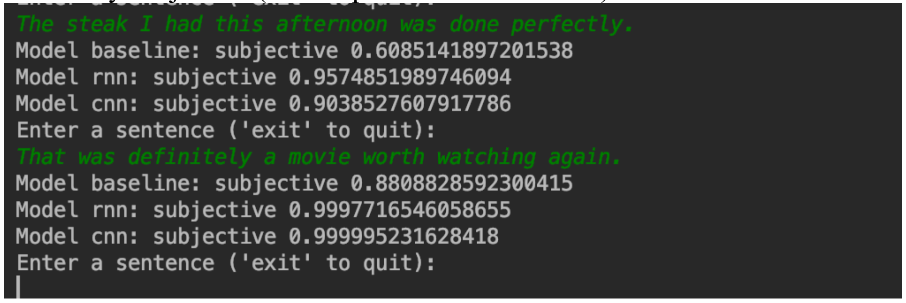
Definitely objective (2 examples for safe measure):



```
/Users/ryando/anaconda3/envs/mie324/bin/python "/Users/ryando/Dropbox/EngSci Fourth Year/MIE324/a4/subjective_bot.py"
Enter a sentence ('exit' to quit):
Justin Bieber is twenty four years old.
Model baseline: objective 0.0441754125058651
Model rnn: objective 0.0005151568911969662
Model cnn: objective 4.244099909556098e-05
Enter a sentence ('exit' to quit):
Christmas is next month!
Model baseline: objective 0.3876916468143463
Model rnn: objective 0.001757791149429977
Model cnn: objective 0.06749600917100906
Enter a sentence ('exit' to quit):
```

For this "definitely objective" case, all models performed exceptionally well, with correct predictions all around at a high certainty as expected. So far the RNN model seems to be performing "the best" with a correct prediction at the highest certainty.

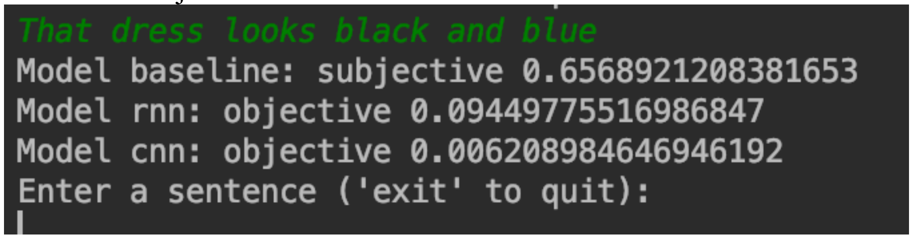
Definitely subjective (2 examples for safe measure):



```
The steak I had this afternoon was done perfectly.
Model baseline: subjective 0.6085141897201538
Model rnn: subjective 0.9574851989746094
Model cnn: subjective 0.9038527607917786
Enter a sentence ('exit' to quit):
That was definitely a movie worth watching again.
Model baseline: subjective 0.8808828592300415
Model rnn: subjective 0.9997716546058655
Model cnn: subjective 0.999995231628418
Enter a sentence ('exit' to quit):
```

For this case, the models again performed very well, behaving as expected with correct predictions all around. Again, the RNN seems to be more polarizing in its correct prediction which leads me to believe that it is performing best.

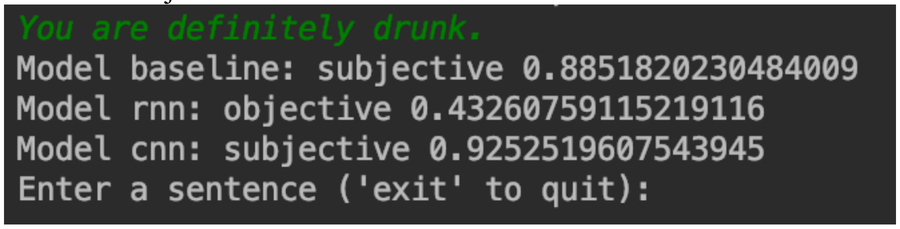
Borderline subjective:



```
That dress looks black and blue
Model baseline: subjective 0.6568921208381653
Model rnn: objective 0.09449775516986847
Model cnn: objective 0.006208984646946192
Enter a sentence ('exit' to quit):
```

I believe this sentence “That dress looks black and blue” is borderline subjective because although it is objective from the speaker’s point of view, it is actually scientifically subjective. The RNN and CNN both confidently predict objective whereas the baseline predicts subjective at a 65% confidence. This is an edge case that incorporates biological subjectivity in visual perception, thereby tricking the model. Surprisingly, the baseline predicts this one correctly over the other two.

Borderline objective:



```
You are definitely drunk.
Model baseline: subjective 0.8851820230484009
Model rnn: objective 0.43260759115219116
Model cnn: subjective 0.9252519607543945
Enter a sentence ('exit' to quit):
```

This statement is a confident observation that someone is drunk. This is a mostly objective statement because identifying someone as drunk is generally undisputed. However, someone could just be acting drunk or is just slightly intoxicated therefore this is *borderline* objective. The RNN is the only one that predicts this correctly.

Overall, out the four cases, the RNN seemed to perform the best. For the “definite” cases, all models agreed with each other. For the others, the models didn’t perform as well (expected, as these cases were designed to trick the system). The RNN seems to work for all cases with the exception of a case where subjectivity is caused by some ambiguity that is biological in origin.

6. *(5 points) Describe your experience with Assignment 4:*
   a. *How much time did you spend on Assignment 4?*

      I spent around 24 hours on assignment 4. It probably would have been faster with more effective use of time and more explicit guidance on implementing the CNN and RNN.

   b. *What did you find challenging?*

      Figuring out the dimensions of all the layer inputs and outputs from the minimal specifications outlined in the assignment handout was challenging and required extensive use of the debugger/ multi-dimensional thinking. This is good practice for future tasks but for an assignment it could have went smoother in this sense (although it worked out at the end).

c. *What did you enjoy?*

I enjoyed being able to use the final model to classify any given sentence into subjective or objective. NLP is a very cool and practical application of neural nets and seeing my completed model in action was rewarding. The process of coding the pre-processing scripts and model implementation also gave me a lot of detailed insight into NLP and implementation experience that I appreciate having.

d. *What did you find confusing?*

For the CNN structure, it wasn't clear at all from the document that there needed to be two convolutional layers in parallel to be concatenated. A more detailed block diagram of the CNN structure could help with this.

e. *What was helpful?*

The instructions along with the PyTorch documentation was generally enough to get through the assignment without extra clarification. Of course, more information would smooth the process but I do understand it's a challenge to balance giving too much information and giving insufficient information. The tutorial at http://anie.me/On-Torchtext/ was also instrumental towards coding the pre-processing scripts using TorchText.