ECE324: Assignment 5
Lisa Li
1003924532

3.1 Create train/validation/test splits

Number of examples of each label:

Label	Training Set	Validation Set	Test Set
1	3200	800	1000
0	3200	800	1000

4/5/6 Overfitting and Full Training Data

For the follow models, all used the same predefined optimizer, learning rate, and loss function as given in the assignment.

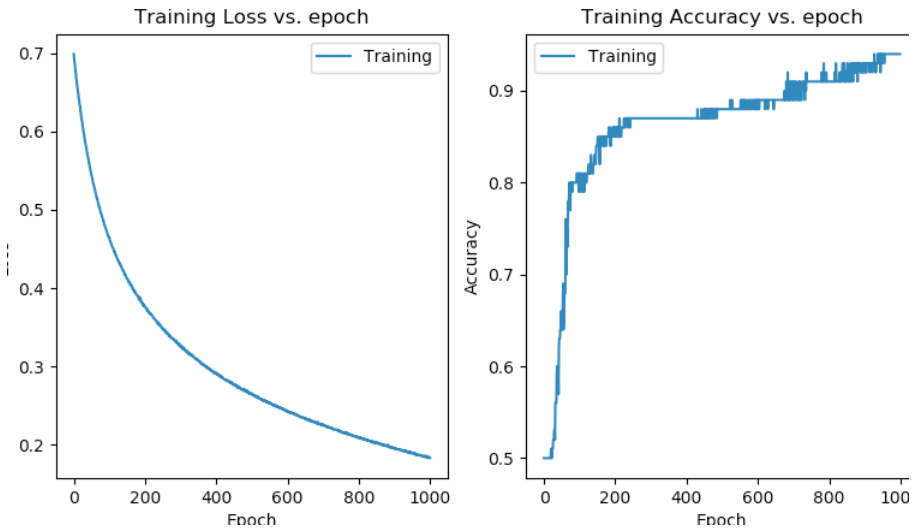
For the overfit iterators:
- A batch size of 5 was used
- 1000 epochs were used for the Baseline, and 100 epochs were used for the CNN and RNN

For the full training:
- A batch size of 64 was used (as given in the assignment)
- 100 epochs were used for the 3 models

The following plots are provided below, with the values of the training/validation/test accuracy and loss provided in Section 8 Question 1.

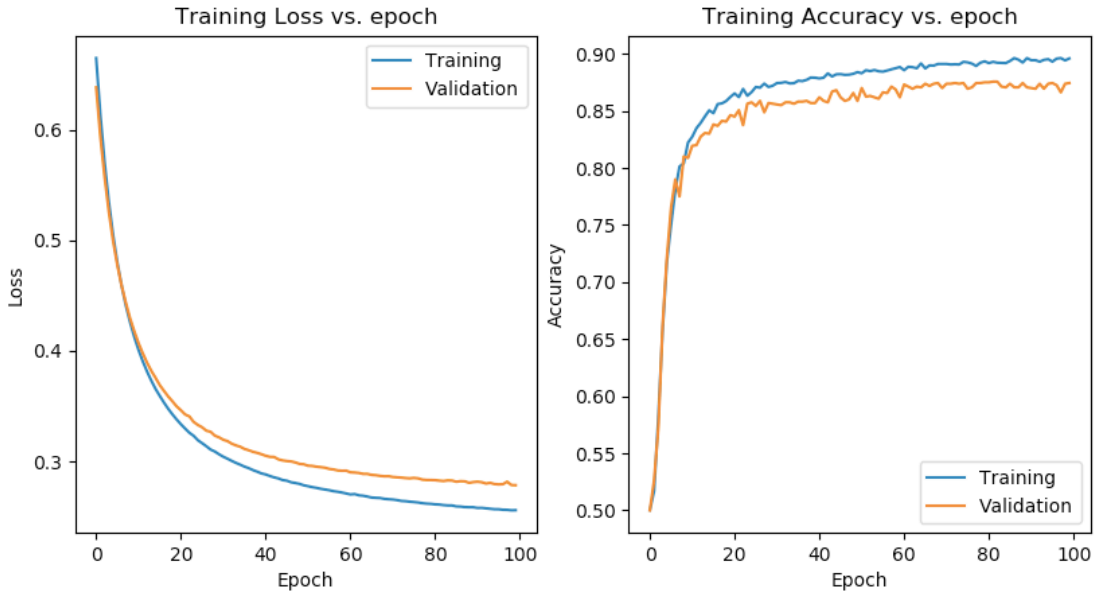
BASELINE

OVERFITTING



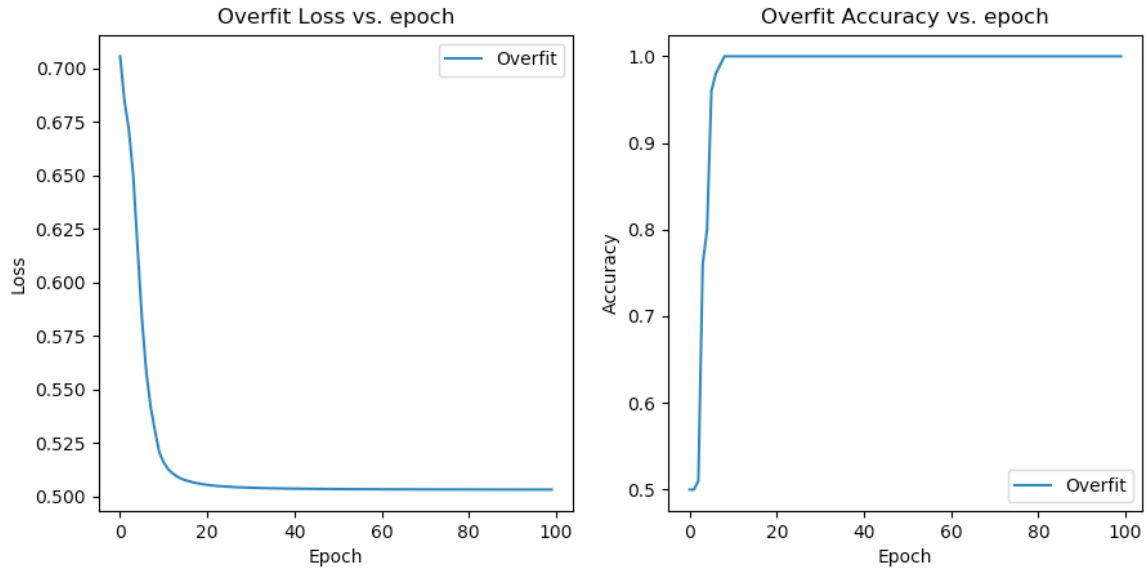
Overfit accuracy: 0.940 | Overfit loss: 0.183

FULL TRAINING DATA



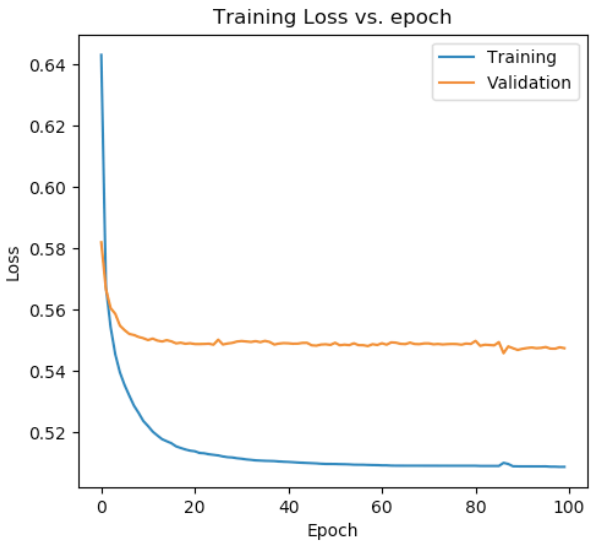
CNN

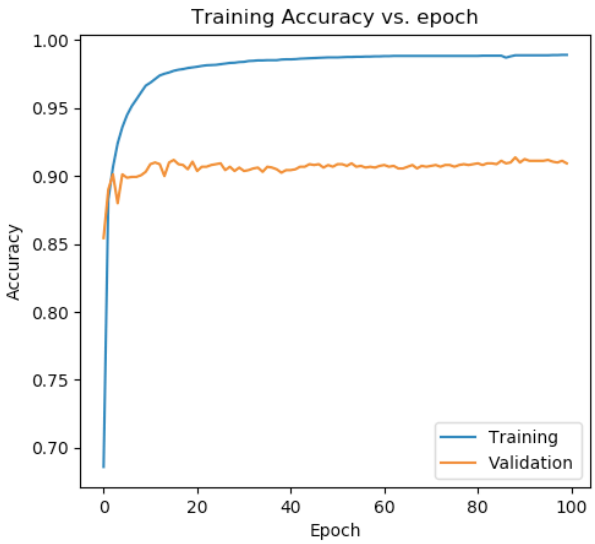
OVERFITTING



Overfit acc: 1.00 | Overfit loss: 0.503

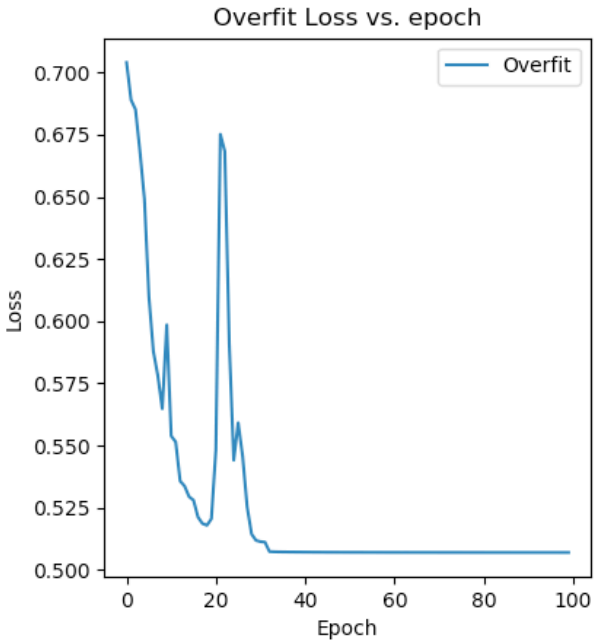
FULL TRAINING DATA

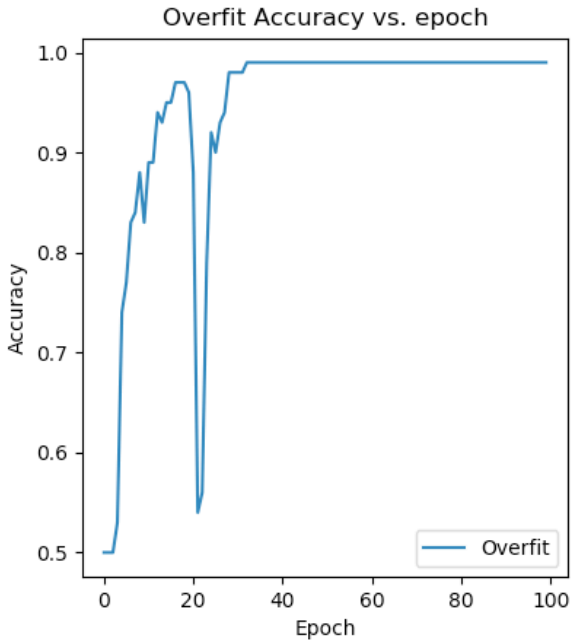




RNN

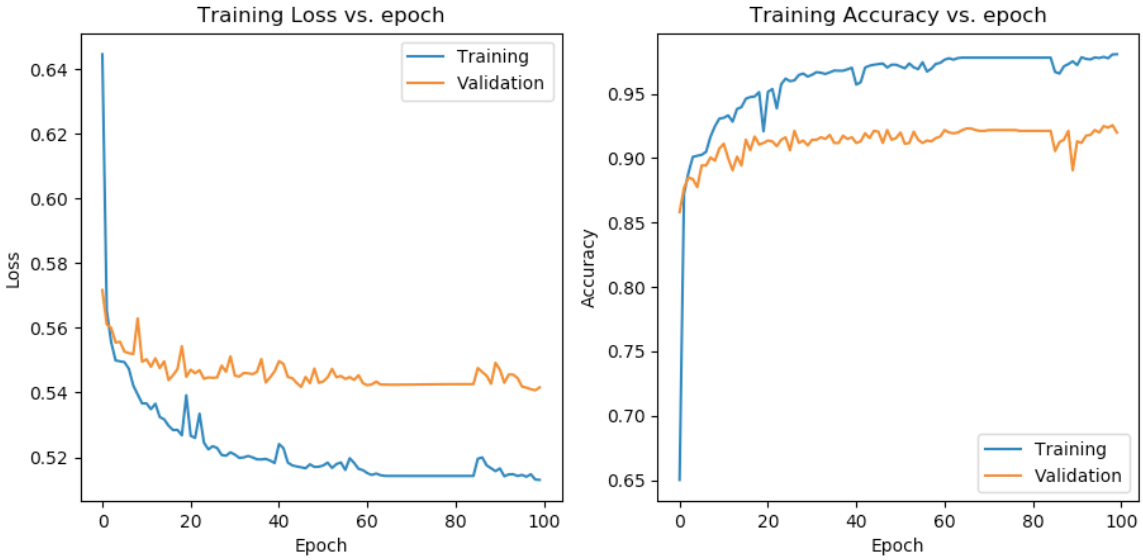
OVERFITTING





Overfit acc: 1.00 | Overfit loss: 0.503

FULL TRAINING DATA



8 Experimental and Conceptual Questions

1.

	Training		Validation		Test	
	Accuracy	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**
Baseline	0.897	0.254	0.876	0.277	0.891	0.278
CNN	0.989	0.508	0.913	0.545	0.913	0.550
RNN	0.980	0.513	0.925	0.540	0.915	0.550

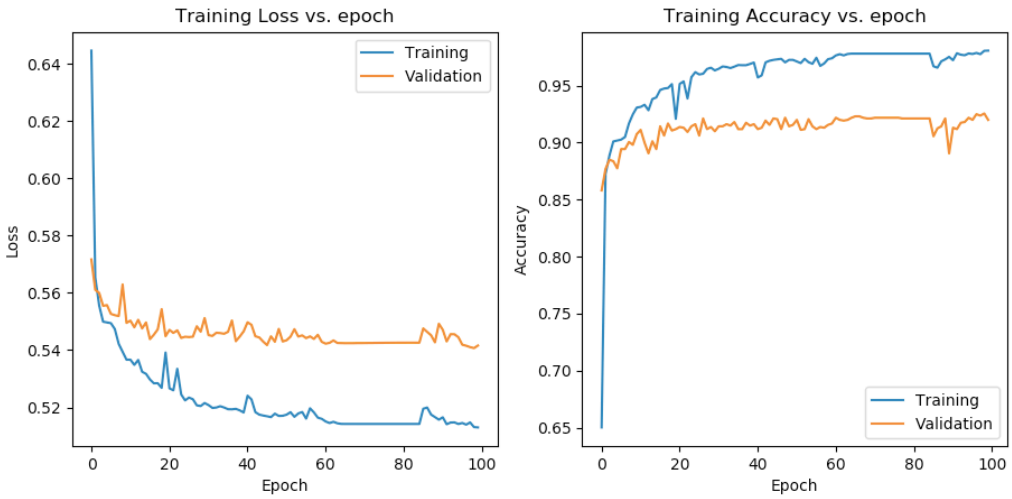
The RNN performed the best on the test set, although only marginally better than the CNN. The results of validation and test accuracy stayed relatively consistent amongst the three models. This makes sense as they were both trained on data that was not used in the training set. As well, they both use the same evaluate function with the same hyperparameters.

2. As we average the word embeddings in a given sentence to find the 'average' meaning of the entire sentence, we do not consider the order of the words as they appear in the sentence. Thus, sentences with the same words but rearranged would provide the same average. This is not ideal as sentences with similar words may have greatly different meanings, as well, order of words adds meaning. If the model performs poorly, it means that the order of the words is important and should not be ignored.

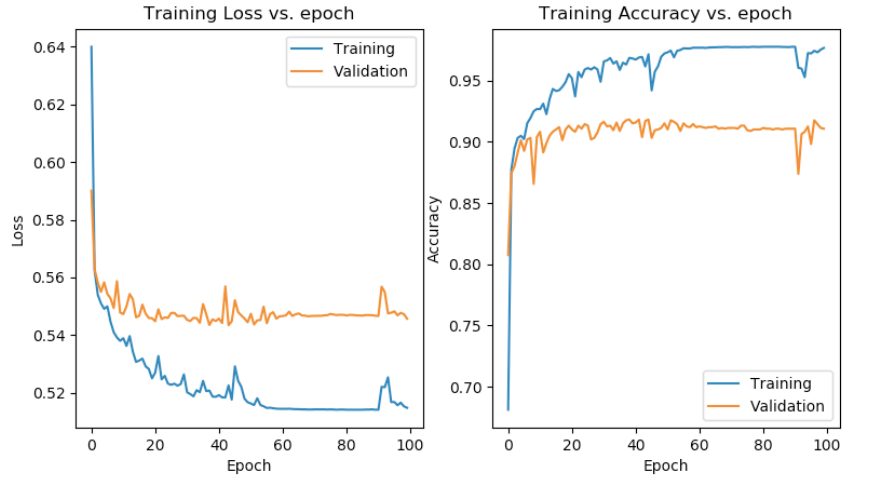
3.

(a) Default scenario, with using pack padded sequence and using the BucketIterator

(Same as Section 6)

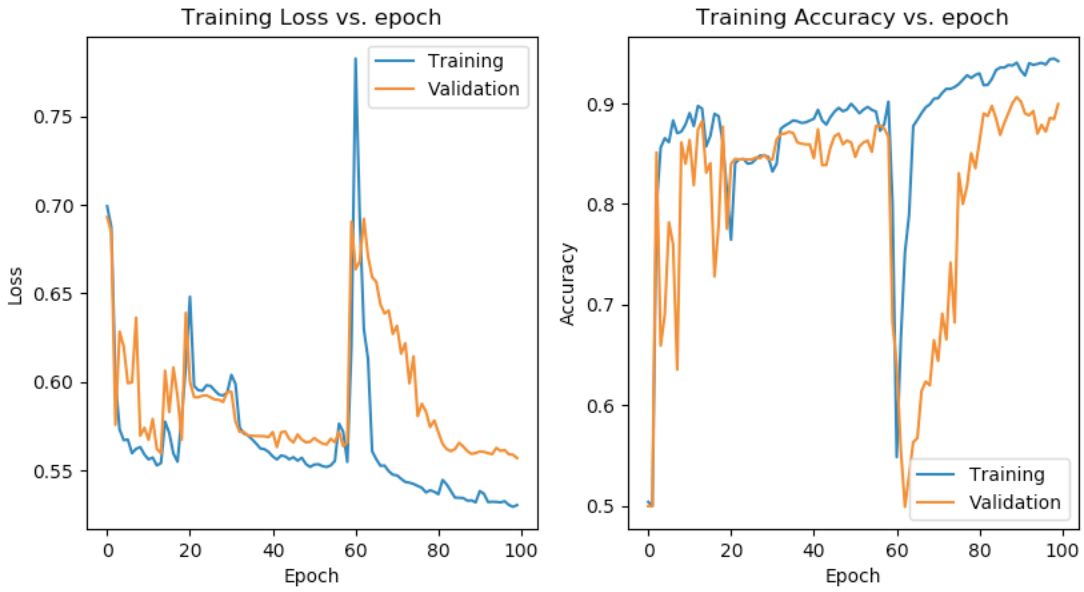


(b) Without calling pack padded sequence, and using the BucketIterator



(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?

Without calling pack padded sequence, and using the Iterator, we are training on sentences that have different lengths and are unsorted by length of the sentence. This causes the final layer of the RNN to output meaningless values if there are sentences with greatly varying lengths, and thus, giving lower accuracy as seen in the table below. As well, we can see that the plot is more bumpy with varying spikes throughout.



	Training		Validation		Test	
	Accuracy	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**
A	0.980	0.513	0.925	0.540	0.915	0.550
B	0.977	0.514	0.918	0.543	0.915	0.550
C	0.944	0.529	0.906	0.557	0.898	0.567

4. In the CNN, through the use of kernels, we are able to evaluate a word in relation with the words beside it (either 2 words, meaning adjacency, or 4 words, a larger grouping), as thus, order of words create significance. However, when we take the maximum, the significance of the entire sentence is lost as we only consider the portion of the sentence with the highest value, which is the most subjective part.

5.

Subjective sentences:

1. Canada has a population of 37 million people (definitely objective)

   Model baseline: objective (0.053)
   Model cnn: objective (0.104)

Model rnn: objective (0.000)

2. Beyonce is the best singer in history (definitely subjective)

Model baseline: subjective (0.986)
Model cnn: subjective (0.653)
Model rnn: subjective (1.000)

3. I think I can cook dinner tonight for you (borderline subjective/objective)

Model baseline: objective (0.464)
Model cnn: objective (0.054)
Model rnn: objective (0.354)

4. I hope I can finish this assignment on time (borderline subjective/objective)

Model baseline: subjective (0.635)
Model cnn: objective (0.475)
Model rnn: objective (0.166)

In the first 3 examples, the models all agreed with each other. In the last one, Baseline and CNN both correctly identified it around 0.5 (borderline), but the RNN rated it very close to the objective side. Overall, Baseline performed the best, with a value near 0 for definitely objective, a value near 1 for definitely subjective, and around 0.5 for the borderline examples. Overall for CNN and RNN, the model correctly predicted subjective/objective, but its values were not respective to the expected results of definite or borderline sentences.

6.

A. A very rough estimate would be 30 hours.

B. Understanding the concepts behind the CNN and RNN. As well, unclear prompts in the write up that led to me assuming a lot of stuff. For example, in section 4.6, did you mean “highest validation accuracy” instead of "lowest validation error”; section 8 question 1, did you mean “report … in a table” instead of “report … in a total”. As well, lots of occurrences where the assignment numbers were messed. Interpreting the assignment caused a lot of debate amongst students, which took up a lot of time, as well as re-training models because of misinterpretation.

C. Learning how to handle word vectors, which will be useful in understanding natural language processing, a very popular topic I’m interested in learning more about and working with.

D. Understanding the in-depth concept of CNNs and RNNs for word vectors. I feel like they were covered very vaguely in class, so creating the models were difficult as it was the first time we dealt with new functions like GRU and embedding.

E. I appreciated the starting template provided for us and its in-depth explanation of the functions. As well, I appreciated how the “outline” of the model was given through text and explanation, including function names and parameter values, and we had to convert it to code ourselves.