Assignment 4

Cong Yu Fang
1003145349

November 9, 2018

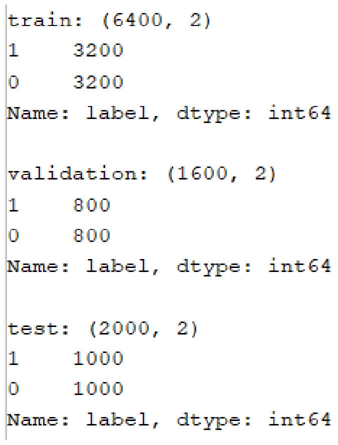
3 Preparing the data

3.1 Create train/validation/test splits

Train: 3200 objective, 3200 subjective, total: 6400

Validation: 800 objective, 800 subjective, total: 1600

Test: 1000 objective, 1000 subjective, total: 2000



```
train: (6400, 2)
1    3200
0    3200
Name: label, dtype: int64

validation: (1600, 2)
1    800
0    800
Name: label, dtype: int64

test: (2000, 2)
1    1000
0    1000
Name: label, dtype: int64
```

7 Grading Experimental and Conceptual Questions

1. 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.

Model: Baseline

train accuracy: 0.885625, train loss: 0.3252662718296051

validation accuracy: 0.8675, validation loss: 0.33857759296894074

test accuracy: 0.8765, test loss: 0.34146514441818

Model: RNN

train accuracy: 1.0, train loss: 0.002222164534032345

validation accuracy c: 0.920625, validation loss: 0.388065107613802

test acc: 0.912, test loss: 0.4547530411509797

Model: CNN

train accuracy: 1.0, train loss: 0.005955078639090061,

validation accuracy: 0.9125, validation loss: 0.24759695410728455

test acc: 0.917, test loss: 0.2693649845896289

The test accuracy of CNN and RNN models are comparable. CNN achieved slightly higher test accuracy in this trial. (The model with highest accuracy is either CNN or RNN and it might be different in different trials since we cannot set the random seed for BucketIterator). The loss for CNN is smaller than that for RNN which means the result predicted by CNN is closer to the actual value. This means CNN would be better than RNN. The difference between validation and test accuracy is really small (within 1%) for all models. It means the algorithm is able to generalize. The validation and test set achieved same levels of accuracy and the minor different between them is simply because one set might have more 'ambiguous' sentences than the others that the models fail to predict correctly. Another reason for the difference might be that the vocabulary is built with 'train.tsv'. There might be words in the validation and test set that are not present in the training set.

2. 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.

Some words in a sentence might be more important than others. Some words might be a stronger indicator whether the sentence is subjective or objective. By taking the average of all the words in a sentence, we might lose that information since all words are weighted equally. Also, the information about the relative position of the words in a sentence is discarded by taking the average. The test accuracy of baseline model is significantly lower than the test accuracy of the other two models, which informs us that some information is indeed lost.

3. 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

train accuracy: 1.0, train loss: 0.002222164534032345

validation accuracy c: 0.920625, validation loss: 0.388065107613802

test acc: 0.912, test loss: 0.4547530411509797

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

train accuracy: 1.0, train loss: 0.001138458028435707, validation accuracy: 0.91375, validation loss: 0.534983867071569

test accuracy: 0.906, test loss: 0.4904670650139451

(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?

The lengths of the sentences in the batch is not uniform anymore if Iterator is used instead of BucketIterator.

train acc: 0.988125, train loss: 0.04414956644177437,

validation acc: 0.898125, validation loss: 0.2836823281645775

test acc: 0.8745, test loss: 0.3283996884711087

As shown above (note the batch sizes for validation and test set are both 64, which is the same as the batch size for training), the validation and test accuracies of scenario a and b are similar but the accuracies for c is lower than a and b. A possible reason might be that the sentence lengths in a batch if using Iterator vary a lot. Since we did not use pack padded sequence, the last hidden state of the last column might not be the correct one to look for (as described in section 4.3). However, if we used the BucketIterator, it would have sentences of similar lengths in one batch such that the last hidden state is the correct state for most of the time. That's probably the reason that accuracy for scenario b is comparable with scenario a but with slightly lower accuracy and slightly higher losses. Therefore, BucketIterator helps to put sentences with similar lengths in one batch so that the last hidden state is correct for most of the times. Pack padded sequence makes sure to select the correct hidden state rather than simply the last hidden state. If only Iterator is used (without pack padded sequence), the model will always pick the last hidden state, which will be incorrect for most cases since the lengths vary a lot and this will yield lower accuracies. Note if the batch sizes for validation and test set are 1600 and 2000 (i.e. evaluating the entire set as one batch), scenario b gives the following accuracies and losses:

train acc: 0.9996875, train loss: 0.0028299621772021055

val acc: 0.5, val loss: 4.404538154602051

test acc: 0.4995, test loss: 4.335085391998291

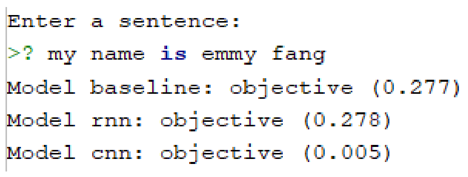
It shows that the train accuracy is high yet the validation or test accuracy is about only 50% and the losses are much higher (about 10 times higher than scenario b with batch sizes of 64). In this case, since we only have one batch, all sentences of various lengths are all in one batch. Therefore, the last hidden state would not be correct for most of the sentences if we eliminate the pack padded sequence, which explains why the low validation and test accuracy.

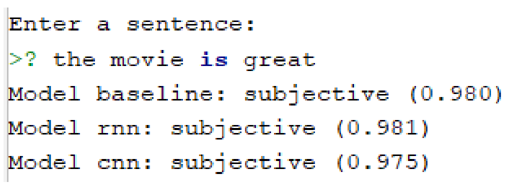
4. In the CNN architecture, what do you think the kernels are learning to detect? When performing max-pooling in the convolution activations, what kind of information is the model discarding? Compare how this is different or similar to the baseline model's discarding of information.

The kernels look for important words or elements of the word vectors in the sentence and assign weights accordingly to form feature maps. Each feature map contains information about a few consecutive words depending on the size of the kernels. In addition, the information about the relative position of words in sentences are retained by feature maps since the kernels are moving along the sentences are creating feature maps along the way. When max-pooling is used, it exacts the largest element from the feature map. Those 'less important' (smaller) values from the feature maps are discarded. It is similar to baseline in the sense that the word vectors are 'averaged' and combined. But CNN is different from the baseline since the baseline weighs all words in the sentences equally by simply taking the average, but the kernels of the CNN weighs different words or different elements of the word vectors differently. Therefore, the CNN will retain more information on the relative importance of the words in comparison to Baseline model. In addition, the max-pooling only exacts the largest value, which discard most of the information on the relative positions of the words like Baseline model does. However, each value from the feature map is calculated from multiple words in the original sentences. Therefore, the CNN will contain more information on positions of the words. This information might be important to understand the meaning of sentences since validation and test accuracy of CNN is higher than that of the baseline model.

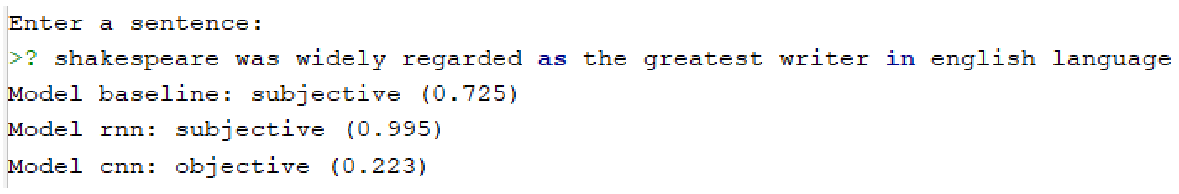
5. Try running the subjective bot.py script on 4 sentences that you come up with yourself, where 2 are definitely objective/subjective, while 2 are borderline subjective/objective, according to your opinion. Include your console output in the write up. Comment on how the three models performed and whether 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?

For the two sentences that are definitely objective and subjective ('my name is emmy fang' and 'the movie is great'), all three models give the correct predictions. The three models agreed with one another. The three models give different predictions for the two borderline sentences. (I would take 'shakespeare was widely regarded as the greatest writer in English language' as objective and 'it is hard to conclude whether he is a good person or not' as subjective.) For the borderline objective case, only CNN gives the expected answer. For the borderline subjective case, only baseline gives the expected answer. In addition to the simple objective sentence mentioned before, the models will give different answers if the objective sentence becomes more complicated ('the most abundant metallic element on the earth is aluminum'). Only CNN gives the expected answer. Therefore, we cannot take the majority vote of the models to be the correct answer. With the five sentences tested, CNN performed the best (4/5 of its predictions are consistent with my expectation).

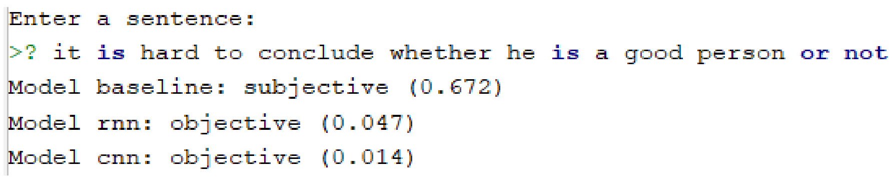

```
Enter a sentence:
>? my name is emmy fang
Model baseline: objective (0.277)
Model rnn: objective (0.278)
Model cnn: objective (0.005)
```



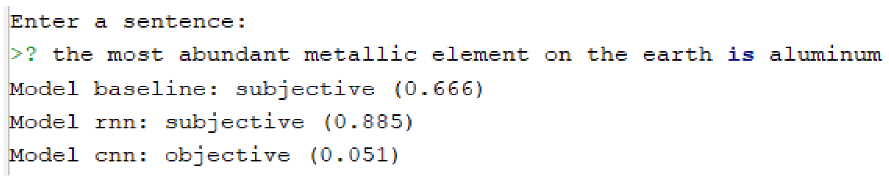
```
Enter a sentence:
>? the movie is great
Model baseline: subjective (0.980)
Model rnn: subjective (0.981)
Model cnn: subjective (0.975)
```



```
Enter a sentence:
>? shakespeare was widely regarded as the greatest writer in english language
Model baseline: subjective (0.725)
Model rnn: subjective (0.995)
Model cnn: objective (0.223)
```



```
Enter a sentence:
>? it is hard to conclude whether he is a good person or not
Model baseline: subjective (0.672)
Model rnn: objective (0.047)
Model cnn: objective (0.014)
```



```
Enter a sentence:
>? the most abundant metallic element on the earth is aluminum
Model baseline: subjective (0.666)
Model rnn: subjective (0.885)
Model cnn: objective (0.051)
```

6.

a) I spent about 30 hours

b) Understanding the documentation of the packages.

c) Figure out how to use those new packages.

d) The instruction for the CNN model took me a long time to understand what to do. More details about the kernel size of [2,4] would be helpful.

e) The references are helpful. The link under section 3.2 on torchtext is really helpful.