CS 489/698 Machine Learning

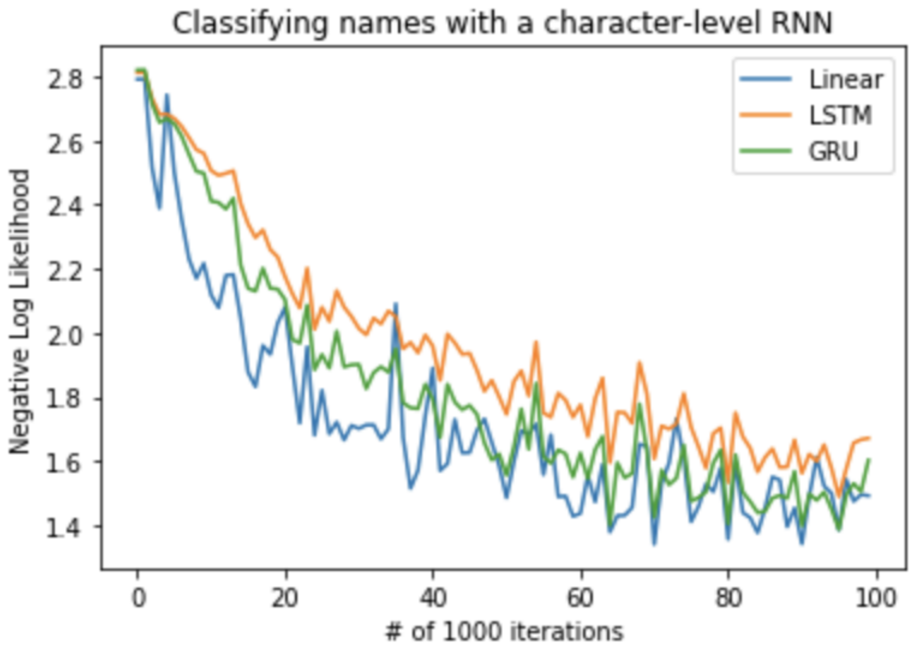
Assignment 5 Solutions

August 14, 2019

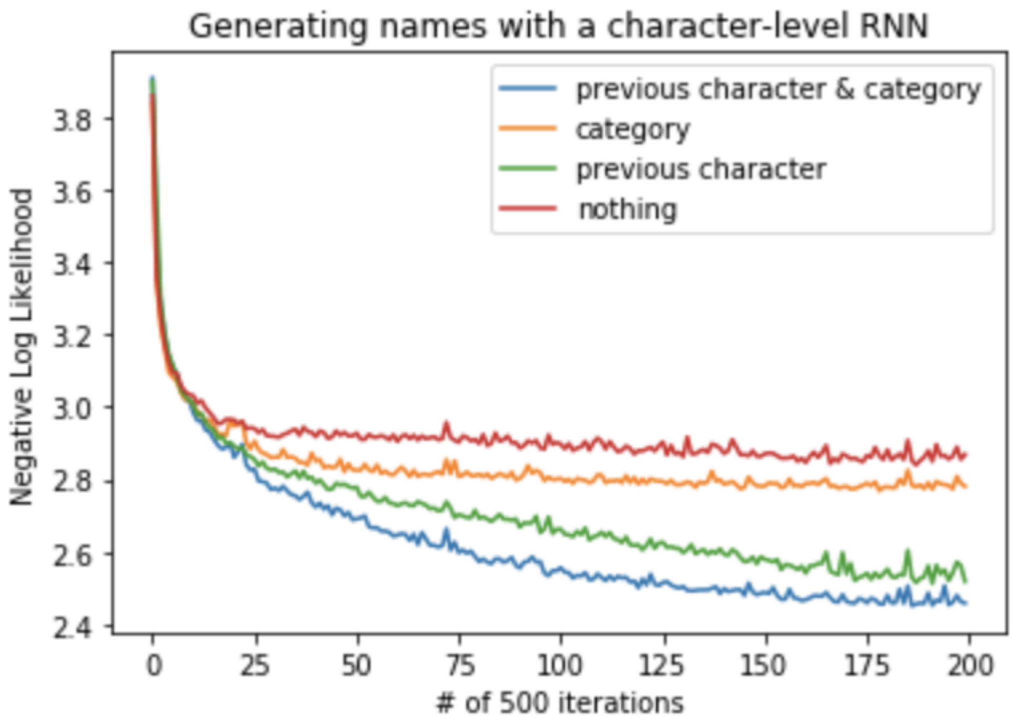
The total grade of this assignment is out of 75 since question 2b is optional and therefore the 25 points associated with question 2b constitute a bonus.

1. [**50 pts**]

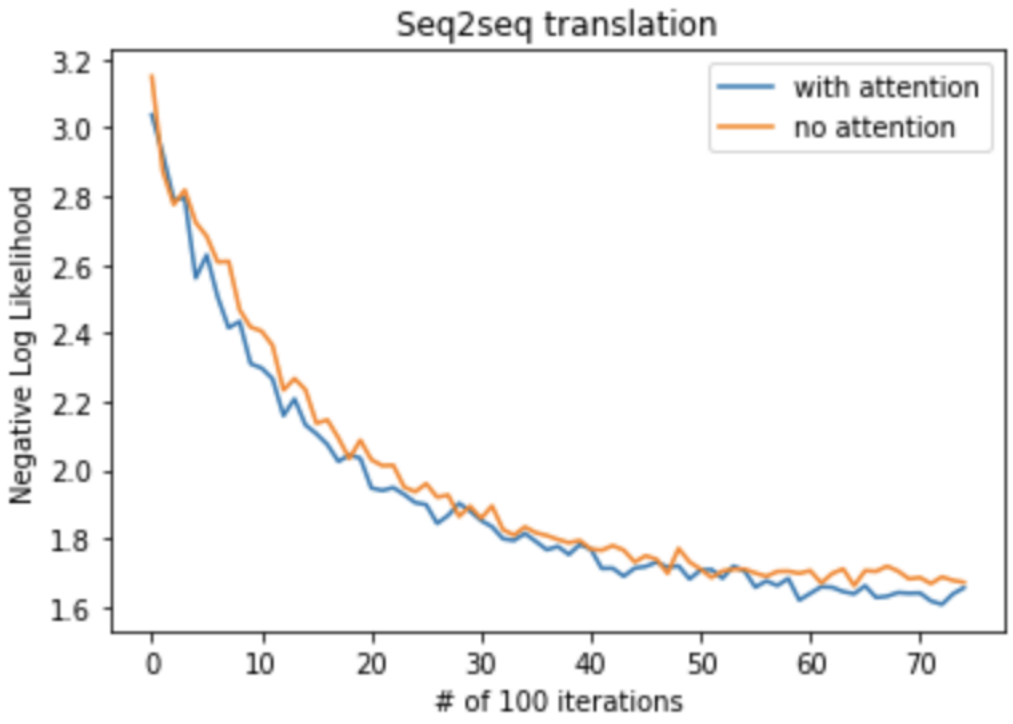
   (a) [**20 pts**] In theory LSTM and GRU units should perform better than linear units since they are more expressive than linear units and they can control the content of their memory by utilizing gates. However, the graph shows that linear units perform better. Since linear units have fewer parameters they need less data, which may explain why they need fewer iterations to find a good model. Similarly, GRU units have fewer parameters than LSTM units and therefore they need less data to find a good model. Another possible explanation is that the optimization of GRU and LSTM units is not as easy as Linear units and therefore they might be more prone to local optima.



(b) **[20 pts]** In theory, it should be sufficient for the RNN to know the category at the start without needing to receive the category as input at every step since it should be able to remember the category. Similarly, in theory, the RNN should not need to receive as input the previous character that it emitted since its hidden state contains the information of the previously emitted character. However, in practice, the RNN would need to learn to remember the category and previous character, which is a difficult task to learn. Hence feeding the category and the previous character as input at every step simplifies the job of the RNN and this regime yields the best results. Since the previous character is much more informative than the category to determine the next character that should be emitted, feeding the previous character only at each step produces better results than feeding only the category at each step. The worse results are obtained when neither the category nor the previous character are fed as input at each step.



(c) **[10 pts]** The results with attention are slightly better than the results without attention. Attention helps the decoder to align its outputs with the inputs. This is especially important for long sentences since the encoder may forget about early tokens. Hence, attention can mitigate the forgetting phenomenon of RNNs and help the decoder look back at the inputs that are relevant for the next outputs.

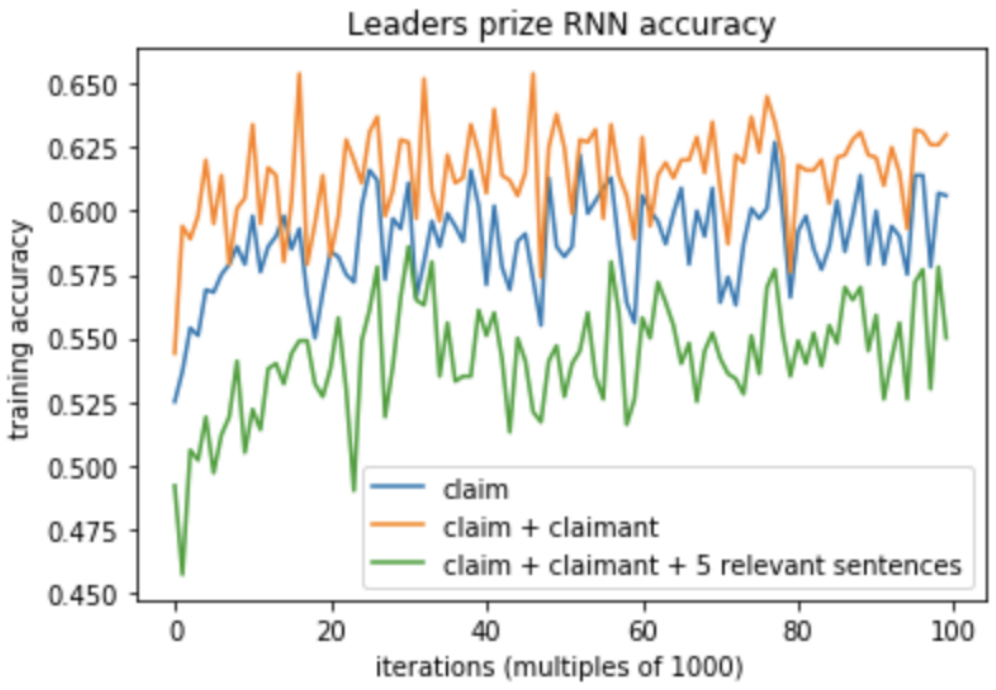


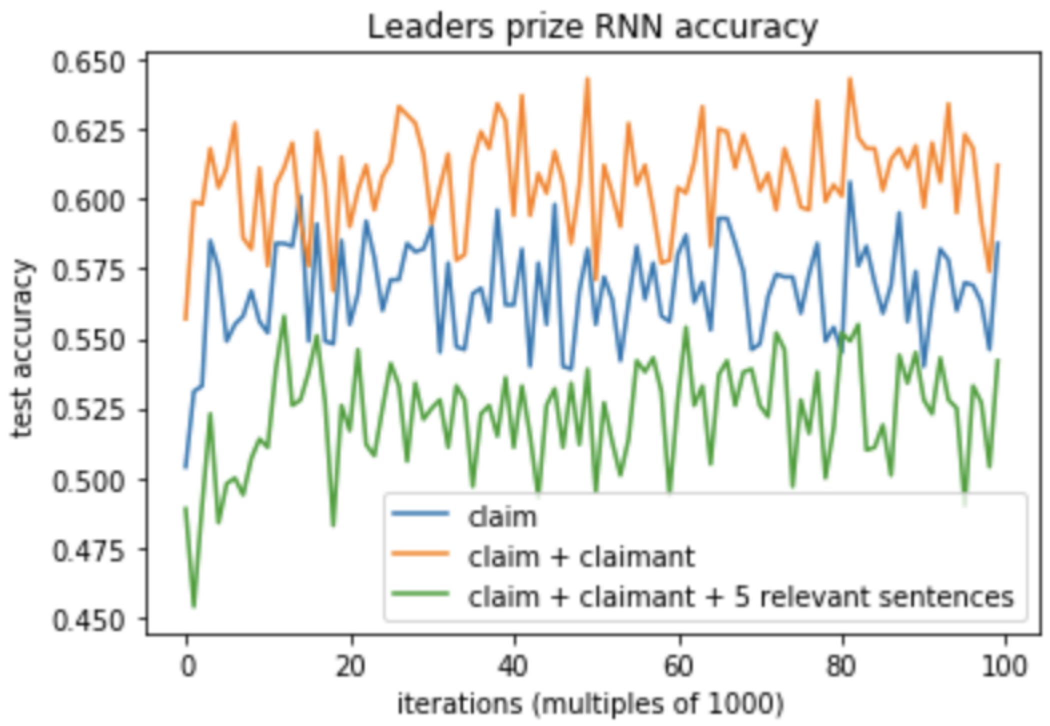
2. **[50 pts]**

   (a) **[25 pts]**

   Note that fake news detection is a notoriously difficult problem (which is why there is a competition with a $1M prize). In the test set, 47.2% of the instances are false, 42.4% are partly true and 10.4% are true. Hence, a simple baseline consists of predicting that all claims are false, which yields an accuracy of 47.2%. The graphs below show that the RNNs improve upon this baseline. Note that the order in which the claim, claimant and 5 relevant sentences are concatenated in the sequence affects the results. The results reported are for the following concatenation order: claim, claimant, 5 relevant sentences.

   In theory, the more information we feed to the RNN, the better the results should be. However, the best results are achieved by combining the claim and the claimant only. Combining the claim, claimant and 5 relevant sentences yields the worst results. There are several possible explanations. First, it is not clear how informative the 5 relevant sentences are. If those sentences do not contain supporting evidence for the label, they are effectively noise. In theory, the RNN should be able to learn to ignore noise, but this is a difficult task. Second, since RNNs tend to forget information fed earlier in the sequence, the results are sensitive to the order of the items and their degree of relevance. Even though the concatenation of the claim, claimant and 5 relevant sentences contains the claim and claimant, the RNN might forget about those items since the 5 relevant sentences are fed last and RNNs have trouble with long range dependencies.

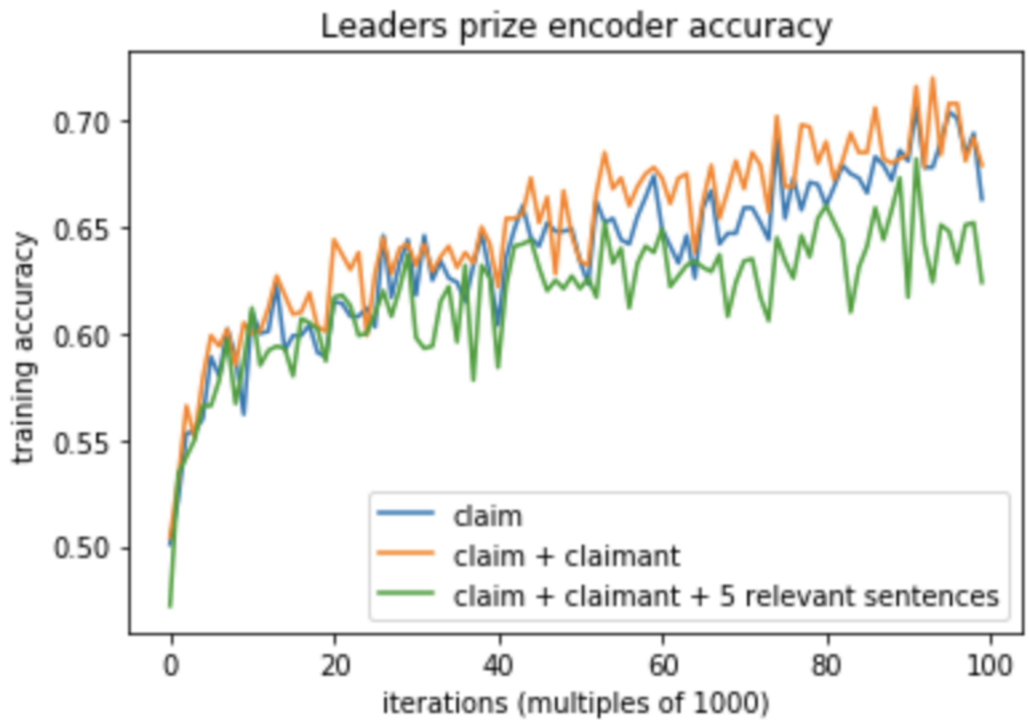


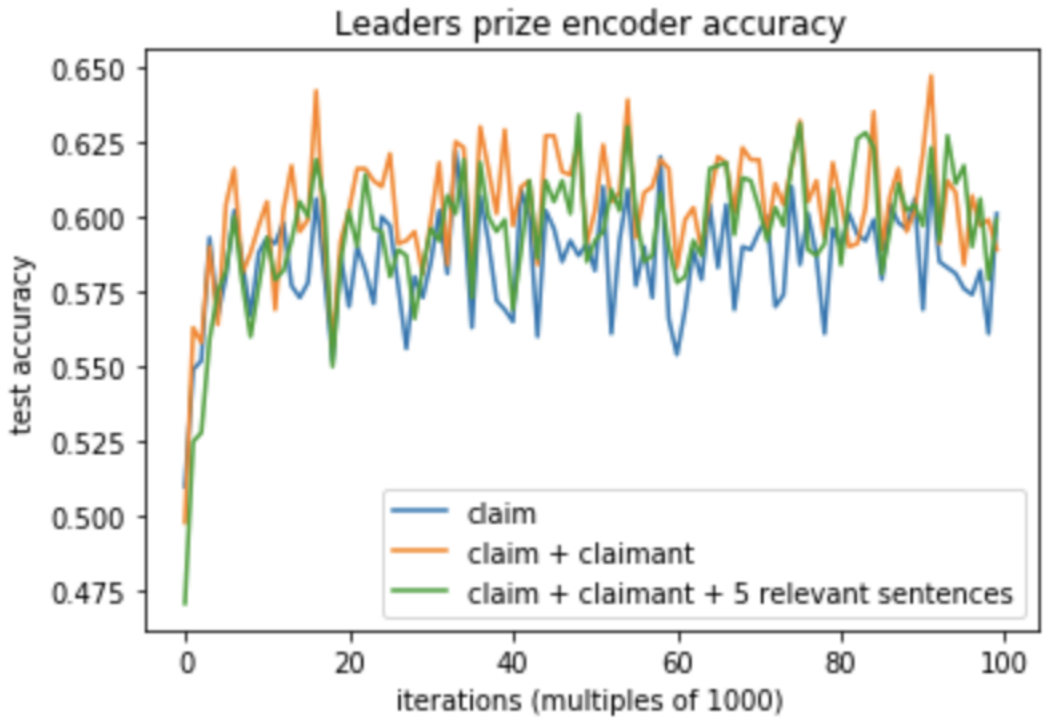


(b) **Optional [25 bonus pts]**

The graphs below show the accuracy for the encoder of a transformer network with a single layer and a single attention head. The output vectors are pooled together with a max operator and then fed to a softmax layer that determines the probability of each label. The same byte pair encoding, Adam optimizer, hyperparameters and number of iterations are used as for the RNN for a fair comparison. The encoder trains slightly faster than the RNN since it processes an entire sequence in parallel. However, when several layers are used, then the training time becomes longer than the RNN, but this is expected since a multi-layer encoder is much larger than a single layer RNN.

The training accuracy of the encoder is better than the training accuracy of the RNN, which indicates that the encoder is more flexible and therefore fits the data better than the RNN. However, the encoder does not generalize significantly better than the RNN for (i) claim and (ii) claim and claimant since the test accuracy is only slightly higher than what was achieved with an RNN. In contrast, for (iii) claim, claimant and 5 relevant sentences, the test accuracy is clearly better for the encoder, which reflects the fact that the encoder does not suffer from the forgetting phenomenon of the RNN since it can attend directly to tokens anywhere in the sequence. In this case, the accuracy for iii) claim, claimant and 5 relevant sentences is higher than for i) claim only, since the encoder learns to leverage the information in claimant regardless of its position in the sequence.


Leaders prize encoder accuracy
training accuracy
0.70
0.65
0.60
0.55
0.50
0
20
40
60
80
100
iterations (multiples of 1000)
claim
claim + claimant
claim + claimant + 5 relevant sentences


Leaders prize encoder accuracy
test accuracy
0.650
0.625
0.600
0.575
0.550
0.525
0.500
0.475
0
20
40
60
80
100
iterations (multiples of 1000)
claim
claim + claimant
claim + claimant + 5 relevant sentences