CSC421: Programming Assignment 3

Due on Thrusday, Mar 22, 2019

Zhongtian Ouyang
1002341012

Part 1

Encoder-Decoder Models and Teacher-Forcing

1.

On long sequences, the architecture in Figure 1 will not perform well. The reason is that the architecture uses a fixed-length vector h_T to represent the input sequence, and h_T is the only information decoder can get. For a long sequence, h_T won't be enough to represent all information of the input sequence. With a information loss, the decoder won't be able to make accurate character-wise predictions. Another reason is that the distance between the first input character and the first output character is too far away.

2.

(a) Increase the length of vector h_T so that it is capable of storing information for long input sequence

(b) Reverse the input sequence.

3.

During training time, the input to the decoder at each time step is the ground-truth token. However, during testing time, the input will be the prediction in the previous time step. Therefore, if one of the predictions is incorrect, all the following predictions are very likely to be incorrect too since they were made based on incorrect inputs. The error accumulates. And the model won't be able to handle this problem because such problem doesn't exist during training.

4.

To address the issue with teacher forcing, we can accomodate the scheduled sampling proposed in the paper. Instead of always using ground-true token during training time, we would randomly choose to use the generated token or the ground-true token during training time. At the beginning, we will have a high probability for using ground-ture token and then gradually decrease such probably. At the end of training, we should mostly be using generated tokens.

Part 2

Gated Recurrent Unit
1.
In nmt.ipynb

2.
The result is not ideal, but OK. For the input "the air conditioning is working", the translated output is "ehtay airway onitingsshay isway ouctingray". We can see some of the words are correct: "airway", "isway". And for the incorrect words, the are somewhat resemble to the correct results: "ehtay" vs "ethay". Also, the model seems to understand the fact that a translated word always end with "ay". It seems like the model do better for shorter words as we would expect.

3.
The combination of "th" always become "ht" when moved to the back.
An incorrect character followed by more incorrect characters.
The first character moved to the back become something else incorrect.
Mess up the charcters in the middle for long words.

Part 3

RNN with Attention
1.
In nmt.ipynb

2.
In nmt.ipynb

3.
The results are overall better than the results from RNN decoder without attention. For shorter words, they are all correctly translated. For the longer words, many of them are correct; most of the incorrect ones only has a minor mistake.
Common Errors:
Missing letters in long words. Messing up at the end of a long word.
Training speed:
In terms of running time, to train for 100 epochs, the model with attention trains much slower, which is the same as the conclusion on the slides. The running time is longer because with attention, we need to calculate the attention context at each time step. Also, there are more weights to train for the model with attention.
In terms of epoch, the validation loss decrease faster for the model with attention.

4.
In nmt.ipynb
Compare to the additive attention, both the results and training speed are very similar. The reason is that when used within the RNN attention decoder, the function of the two attention method are very similar. The only differences are the function used to calculate f(Q, K), and the way context is calculated using attention and value.

Part 4

Attention is All You Need
1.
Additive attention:
Advantage:More flexible
Disadvantage:Needs two sequential steps because there are two layers and we need to calculate them one after another
Scaled dot-product attention:
Advantage:Can attend to multiple entries, more efficient due to matrix multiplication optimizations
Disadvantage:More parameters means slower to train and easier to overfit.

2.
In nmt.ipynb

3.
In nmt.ipynb
Compare to those from the previous decoders, the validation loss and training loss from transformer are much smaller. The actual translated texts are also better in quality. The sentence "the air conditioning is working" is correctly translated. And most of the words I tested are also correctly translated.
In terms of training speed, training transformer decoder for 100 epoches only takes about half of time compare to the previous decoders. Transformer decoder is faster to train as we expected.

4.
When training the model, we can observe that both losses are significantly higher. The results are very bad compare to the ones from causal model. Only very few words are correct, while the others are either blank or repetition of several letters. The reason is that without using causal attention for the self-attention, the model is trained to predict knowing both the previous and following characters. But when we test the decoder, it doesn't know the future. This discrepancy between training time and testing time make the decoder unable to perform well when making real predictions.

5.
Our simple transformer decoder work without explicit positional encoding because the sequence is relatively short in our model and the information about the positions of the input sequence is captured by the encoder annotations.

Part 5

Attention Visualizations

Word: cake → akecay
Correctly translated by both RNN and Transformer

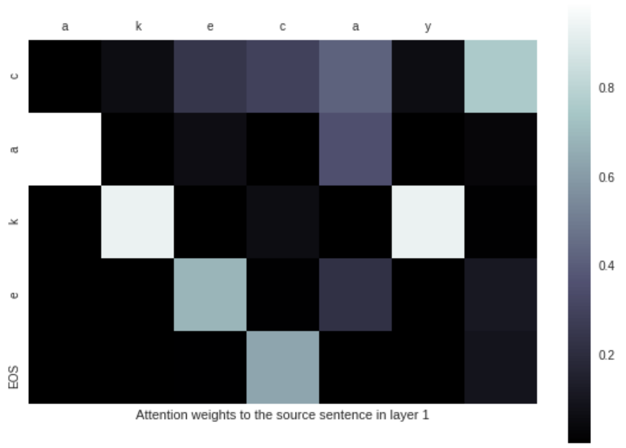
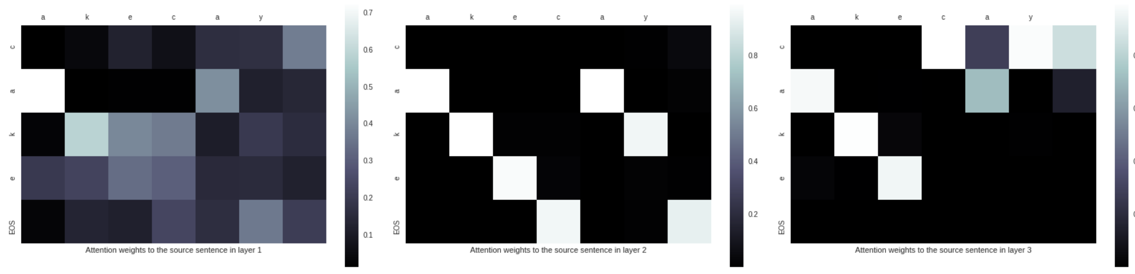


Figure 1: RNN Attention Map



(a) layer 1 (b) layer 2 (c) layer 3

Figure 2: Transformer Attention Maps

This is an example of a success case. We can see that in the rnn attention map , the attention for the characters “a”, “k”, “e” are almost perfect. The letter “c” has a somewhat split attention on “c” and EOS, which is reasonable because “c” should be at the position of EOS with the letter being “c”. Attention for “a”, “y” at the end of the word doesn’t really matters because all the translated words should end with “ay”.
For the transformer decoder, the attention in the first layer is rather scrambled. But in the layer, the attention is perfect. Notice that the attention for letter “c” in layer 2 and layer 3 indicates that the model understand that it should put an extra letter at EOS in this case and the letter should be the first letter.

Word: drink → inkdray
Correctly translated by both RNN and Transformer

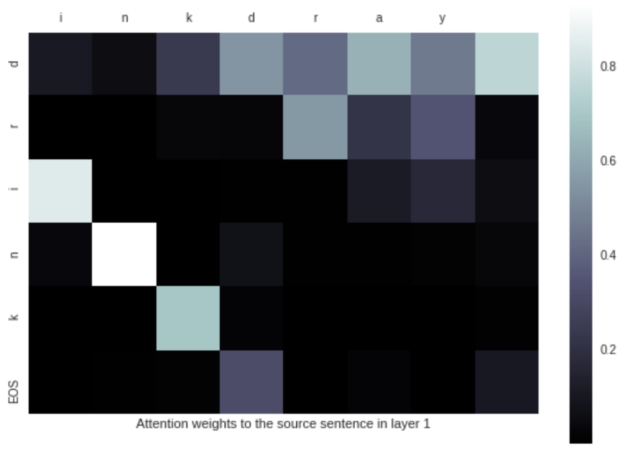
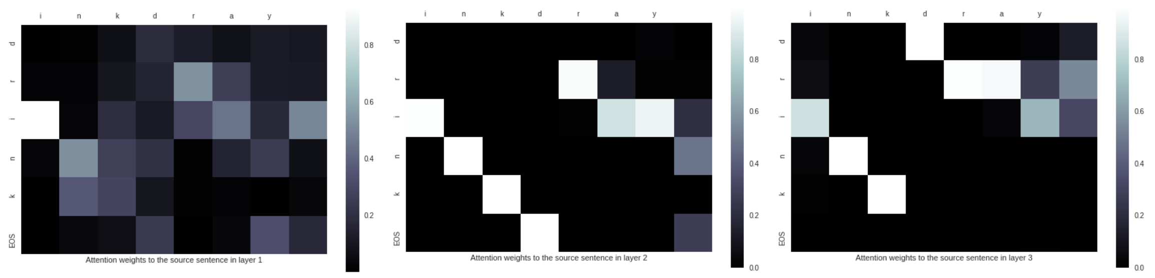


Figure 3: RNN Attention Map



(a) layer 1 (b) layer 2 (c) layer 3

Figure 4: Transformer Attention Maps

This is another success case. The Attention maps for both the RNN and Transformer are very similar to the previous case.

Word: aardvary → aardvaryway
Incorrectly translated by RNN. Correctly translated by Transformer

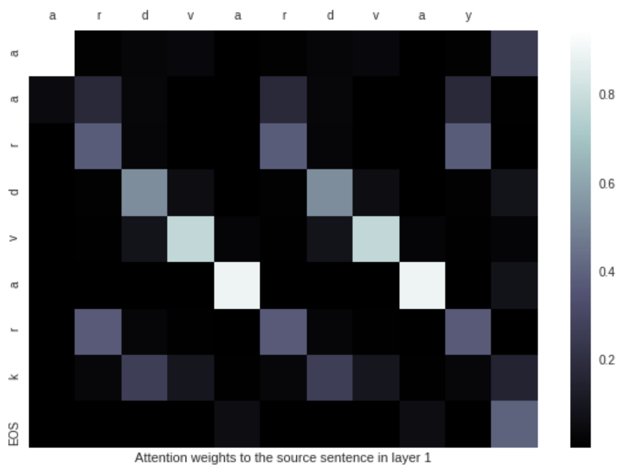
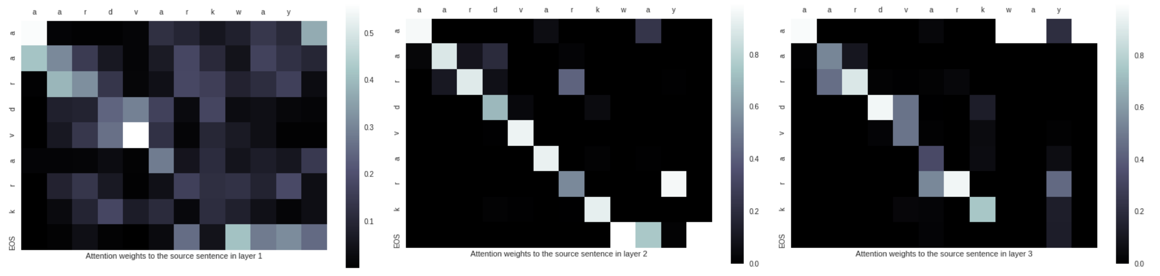


Figure 5: RNN Attention Map



(a) layer 1 (b) layer 2 (c) layer 3

Figure 6: Transformer Attention Maps

Here, the RNN fail to translate the word. Because of the unusual and repetitive structure the word has, the RNN is confused after the second "a". The attention is divided evenly between the two "r" characters, and mistakenly believe this character is the first "r" character instead, which resulted to a "d" instead of "k" and following errors.
As a comparison, we can see that the attention for the second and third layers of Transformer are concentrated on the diagonal. The letter "w" is also correctly added to the translated word by looking at the first letter and identifying it as a vowel.

Word: well-mannered → ellway-anneredmay
Correctly translated by RNN. Incorrectly translated by Transformer

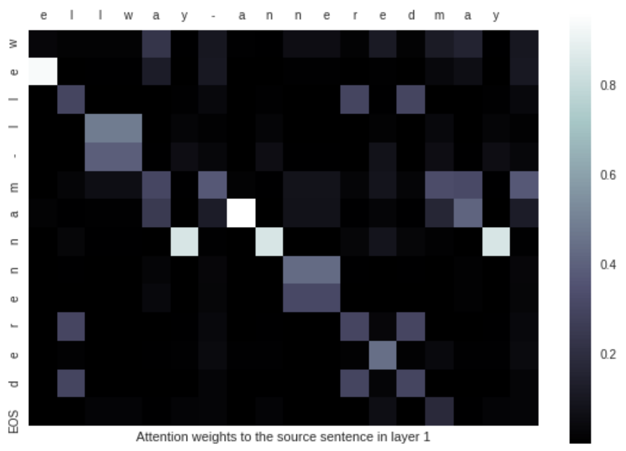
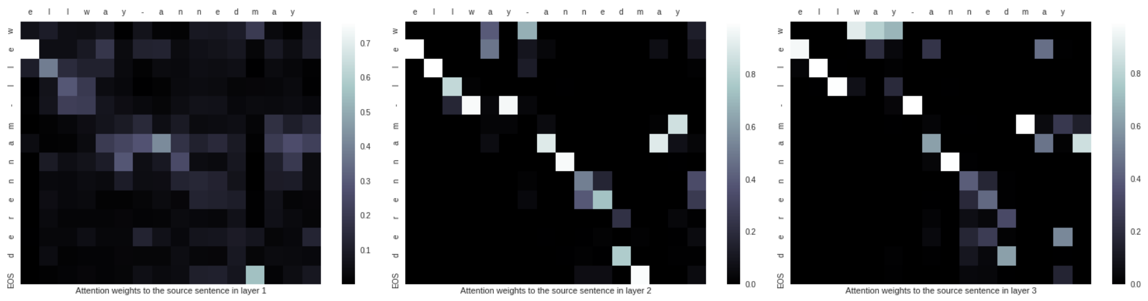


Figure 7: RNN Attention Map



(a) layer 1 (b) layer 2 (c) layer 3

Figure 8: Transformer Attention Maps

Unexpectedly, in this case, RNN make the correct translation while Transformer doesn't. From RNN's attention map, we can see that the attention is in fact not very promising. The fact that it make a correct translation is accompanied with some luck. From Transformer's attention maps, we can see that it is confused after the first "e" in the word "mannered". It mistakenly attends to "d" after the second "e" instead of "r" which is the correct result.

Word: abcddcba → abcddcbaway
Incorrectly translated by RNN and Transformer

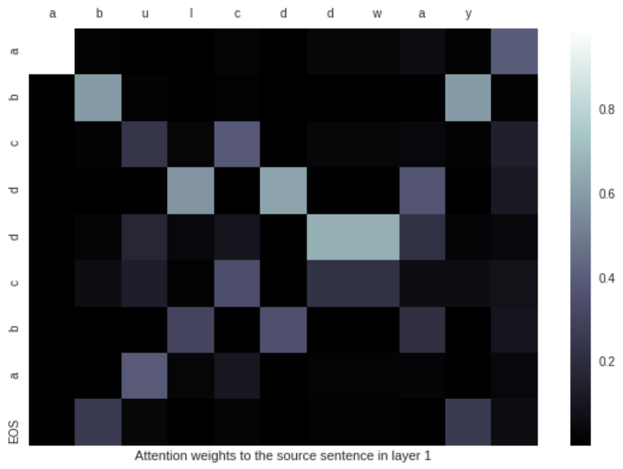
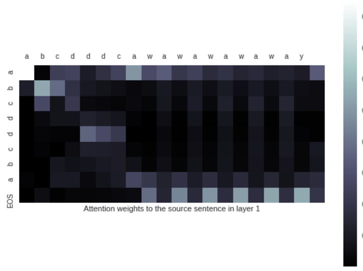
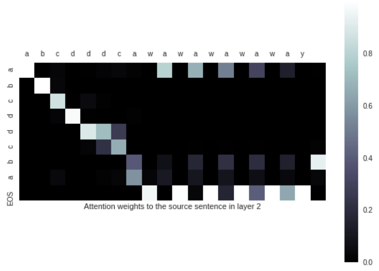


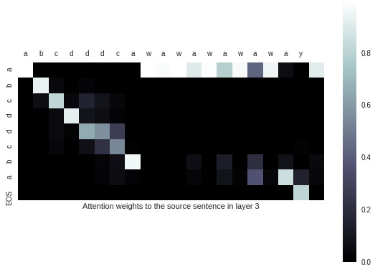
Figure 9: RNN Attention Map



(a) layer 1



(b) layer 2



(c) layer 3

Figure 10: Transformer Attention Maps

Here, we can see the made-up word with unusual character combination and unusual symmetric structure gives both RNN and Transformer a hard time. For the RNN, we can observe that the top half and bottom half of the attention map is also somewhat symmetric. The symmetric structure makes RNN unable to attend to the correct letter and outputting random characters that was not exist in the original word. Transformer did a little bit better. The attention is mostly on the diagonal and the first part of the word is almost correct. However, Transformer seems to be unable to decide when to stop the word. The reason may be that the word is made up and the letter combinations are unusual.