CSC321 A3

Name: Tianyu Zhao (1002109470)

March 2018

Part 1. Encoder-Decoder Models and Capacity

Question 1:

I think this architecture will perform bad on long sequences. Long sequences will convert into the same length as short sequences, since the encoder RNN compresses the input into a fixed-length matrix. This may results losing information when the input is a very long sequences. Furthermore, the weight of the first input character may be very small or very large when the sequence is long.

Question 2:

The result I got from running the script

```
b2240-06:~/CSC321/a3-code$ python translate_no_attn.py
roomba --> omcerway
concert --> orcortcay
hello --> erlehay
table --> adletay
she --> ethay
shopping --> optinnscay
bape --> amesay
ape --> ameray
car --> arpay
cars --> arscay
cat --> atcay
do --> oday
did --> idday
he --> ehay
oops --> ooceway
wonderful --> onleraylway
```

The result is bad. From the result, we can see that the model does better for the short sequence. However, it does not perform correctly even for some short sequences like "car" in the result I provided. It performs worse for the long sequence. Therefore, it is very likely to have bad results for the long sequence word and it is not guaranteed to have correct output for the short sequence.

Part 2. Teacher-Forcing

Question 1:

For the teacher-forcing, the decoder architecture feeds the ground-truth token into the next input during training. This will force our model to learn the correct results. When we are testing the model, the decoder does not have access to the ground-truth output, and it takes the previous output as the next input. This will be bad when the decoder generates the wrong result at the very beginning and the decoder uses the wrong output to be the next input. The output during generation is very unstable.

Question 2:

We can use curriculum learning approach to address this issue. Since we feed the ground-truth token each time, the model can not learn the mistakes during the training. If we let the model learn from their mistakes during training, our model can fix their mistakes during generation or inference.

Part 3. Gate Recurrent Unit (GRU)

Please see the code

Part 4. Implementing Attention

Question 1:

Please see the code

Question 2:

Please see the code

Question 3:

The attention graph:

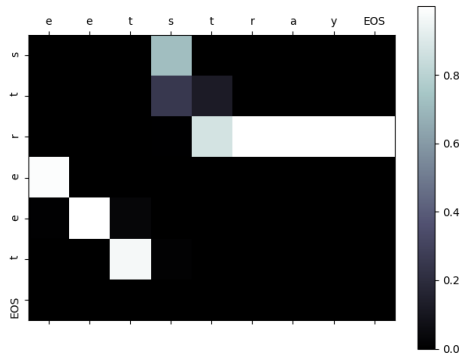

Fig 0.1 attention graph
The loss_plot:

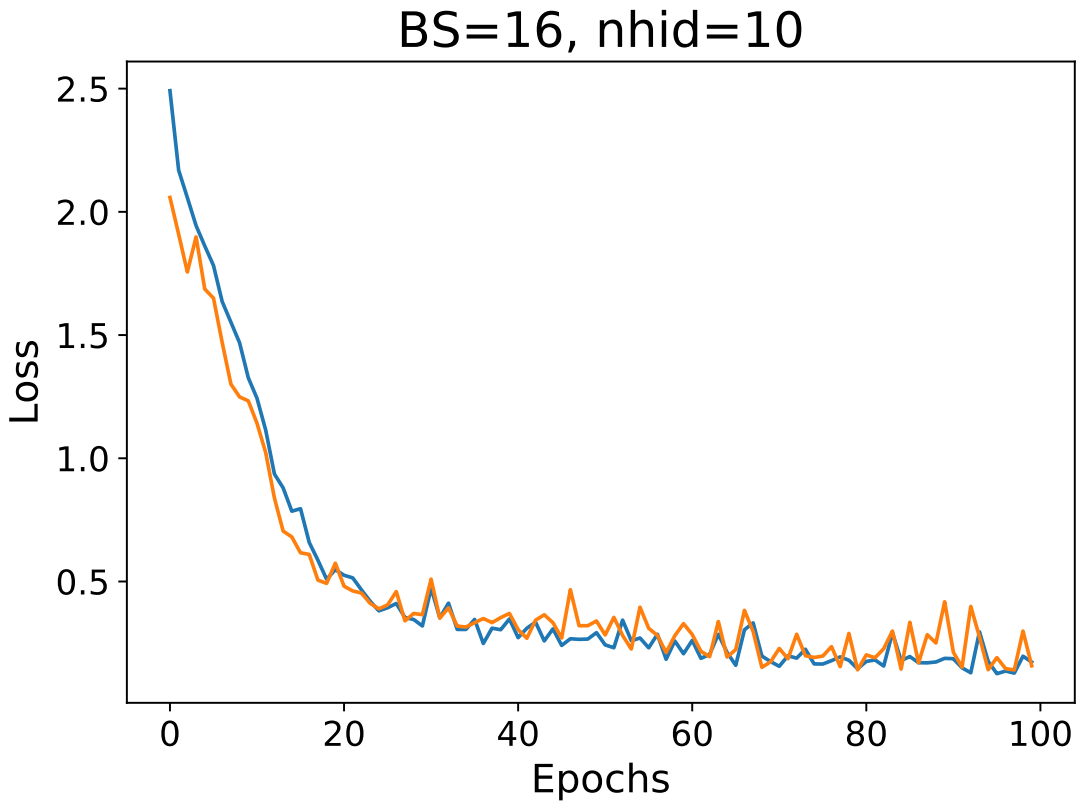

Fig 0.2 loss_plot

Part 5. Attention Visualizations

Question 1:

Result of running the script:

```
b2240-09:~/CSC321/a3-code$ python visualize_attention.py --load checkpoints/h10-bs16/
```

```
roomba --> oombaray
james --> amesjay
cake --> akecay
mathematics --> athematinsmay
-- --> --
chen --> enchay
he --> ehay
aardvark --> aardvarkway
drink --> inkdray
university --> universitiway
architecture --> architectureway
kanqiu --> anqiuway
well-mannered --> ellway-inasasyay
- --> --
tom --> omway
vfde --> escay
floccinaucinihilipilification --> occinnauniniiiioioww
```

From the results, there are 6 cases in total that getting wrong results.

Failure Cases:

```
university --> universitiway
architecture --> architectureway
kanqiu --> anqiuway
well-mannered --> ellway-inasasyay
- --> --
tom --> omway
vfde --> escay
floccinaucinihilipilification --> occinnauniniiiioioww
```

There are multiple reasons that could cause the wrong output. The most obvious cause is long-sequence word. From the above examples, we can find out that it is very likely to generate the wrong result when the length of a word is too long (E.g: university and architecture). Another reason might be patterns that are never occurred in the training example (E.g: '-' , 'kanqiu', 'vfde'). From Fig 1.5, the color of the diagonal goes from white to color close to black. This means that our model is less confident for characters that are in the end of the words. The first three words are correct since the patch color is white, and the color goes to black gradually. We can expect that the output for the latter characters are false.
Take 'vfde' as an example. In English, there is no word starting with vf which means the pattern 'vf' is new and fresh. The model does not recognize the pattern successfully and the result is wrong as we expected. From Fig 1.1, the upper part of the graph is all black which means that the model is not confident with the output. Actually, it is 100% not confident. This explains why the output is totally wrong.

Also, let's check the word 'well-mannered'. From Fig 1.2, we can see that the probability at the beginning of the word is very high, and it drops dramatically since '-'. Since 'well' is a normal English word and it actually exists a lot of times in our data set, our model successfully identifies the pattern and produces the correct result for 'well'. However, the second part 'mannered' generates a very bad result which might because of the unfamiliar pattern. Furthermore, The lowest probability occurs close to the end of sentence. This might caused by long-sequence. As the word 'well-mannered' is a long sequence word, we can expect the result is bad and the result we got actually prove this point.

The output of '–' is correct since '–' is in our training data set, and when we want to get the output of '-', the model produces a wrong output. '-' usually occurs in the middle of the words and since we give single '-' as an input, the model does not recognize the pattern.

Success Cases:

```
roomba --> oombaray
james --> amesjay
cake --> akecay
mathematics --> athematinsmay
-- --> --
chen --> enchay
he --> ehay
aardvark --> aardvarkway
drink --> inkdray
```

From Fig 1.3 and Fig 1.4, we can see that the diagonal of the graph is white or close to white. This means that our model is very confident to choose the correct output. The top-right corner is white because the model successfully identifies the pattern and knows the correct letters to be added in the end. Even though the word 'mathematics' is long, the model produces the correct output since it is a normal English word and follows the general rules.

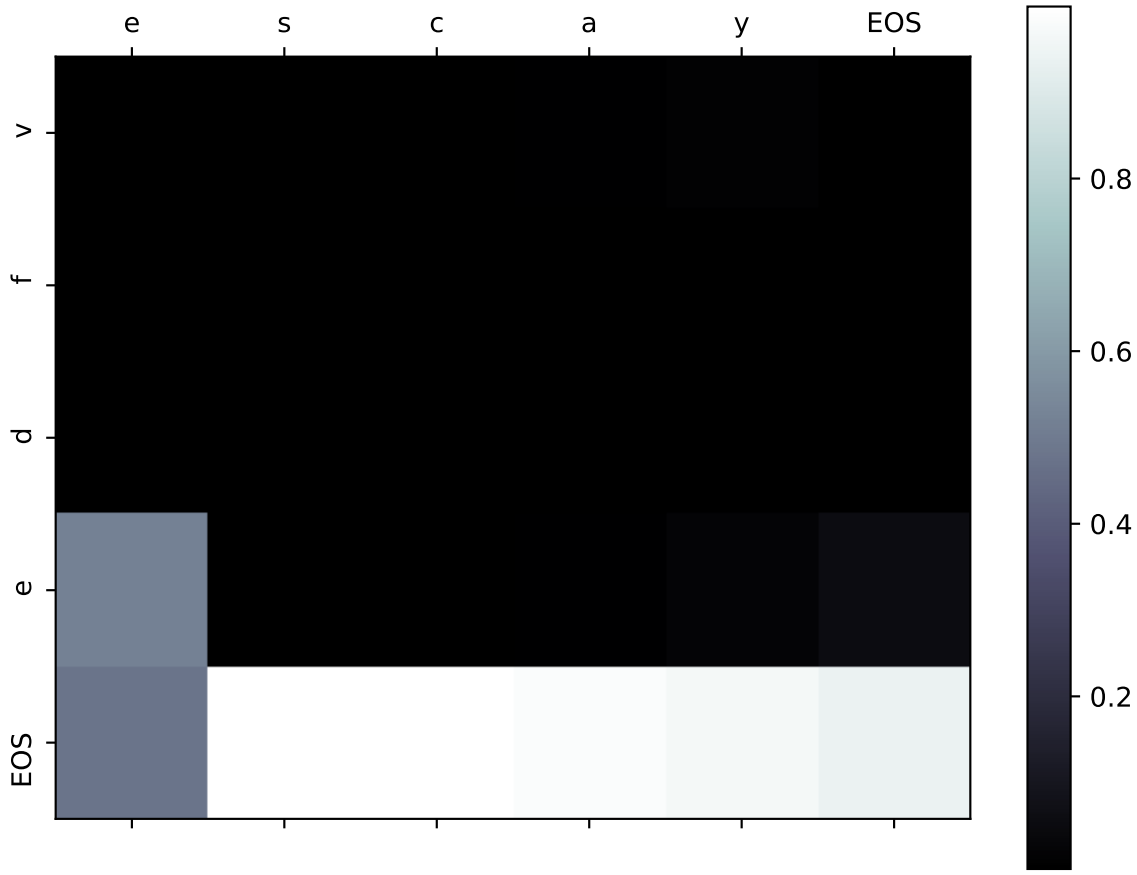

Fig 1.1 vfde

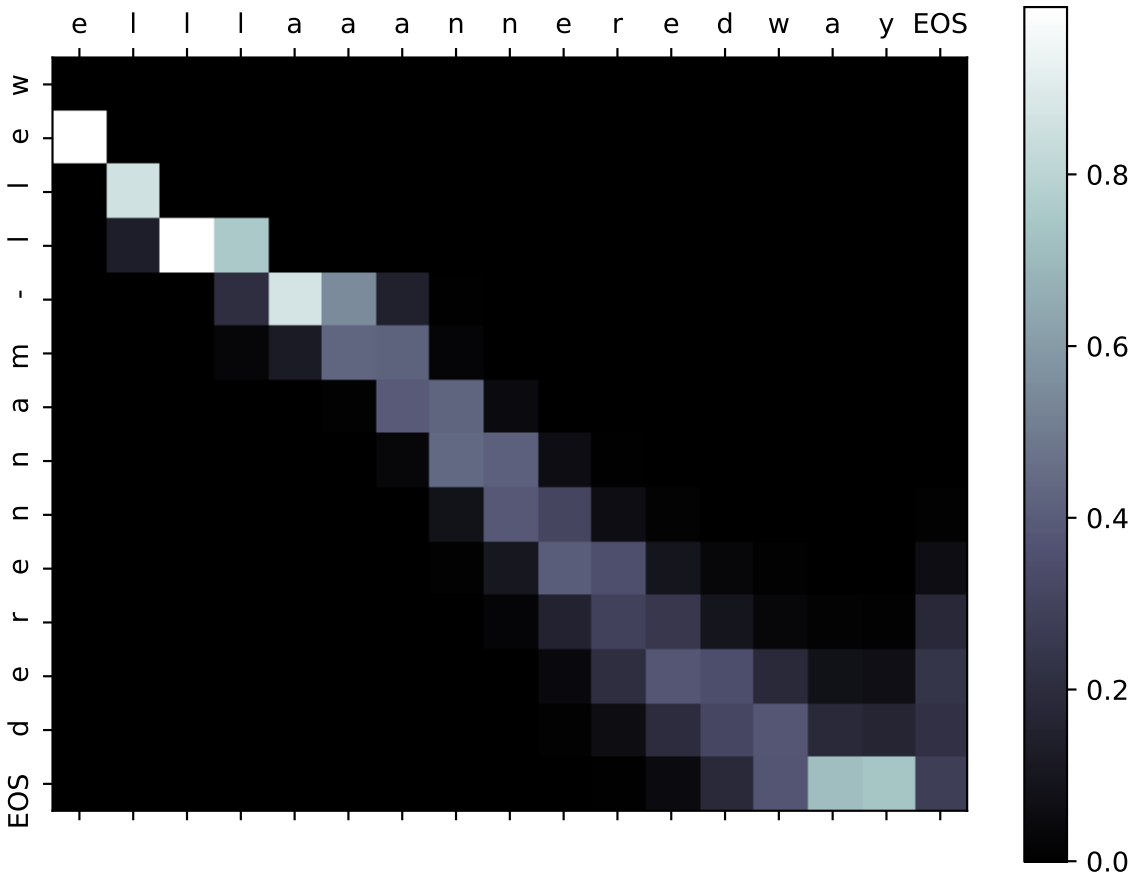

Fig 1.2 well-mannered

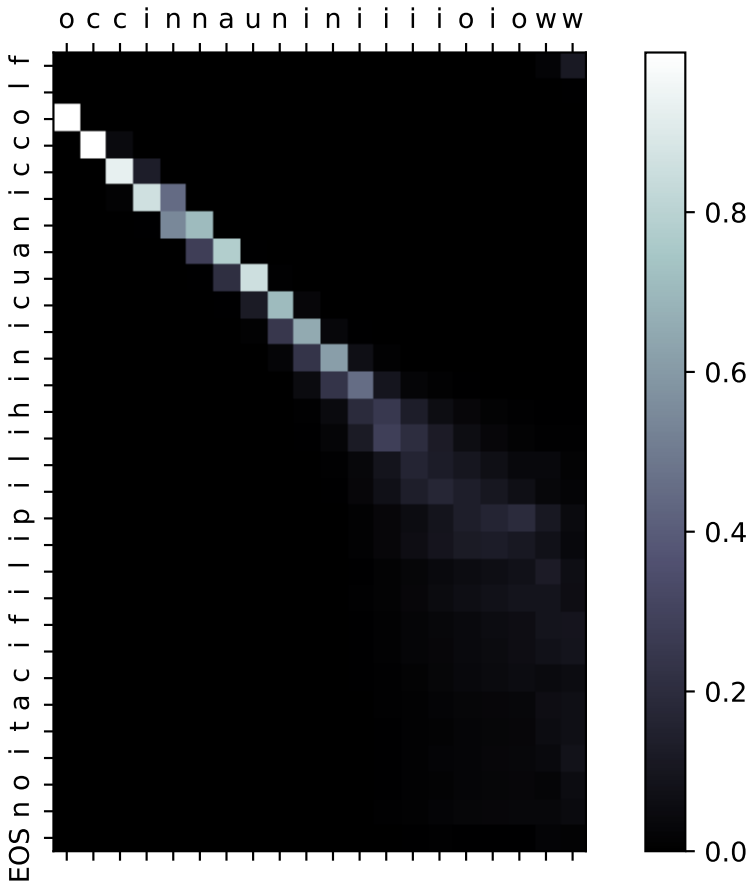

Fig 1.5 floccinaucinihilipilification

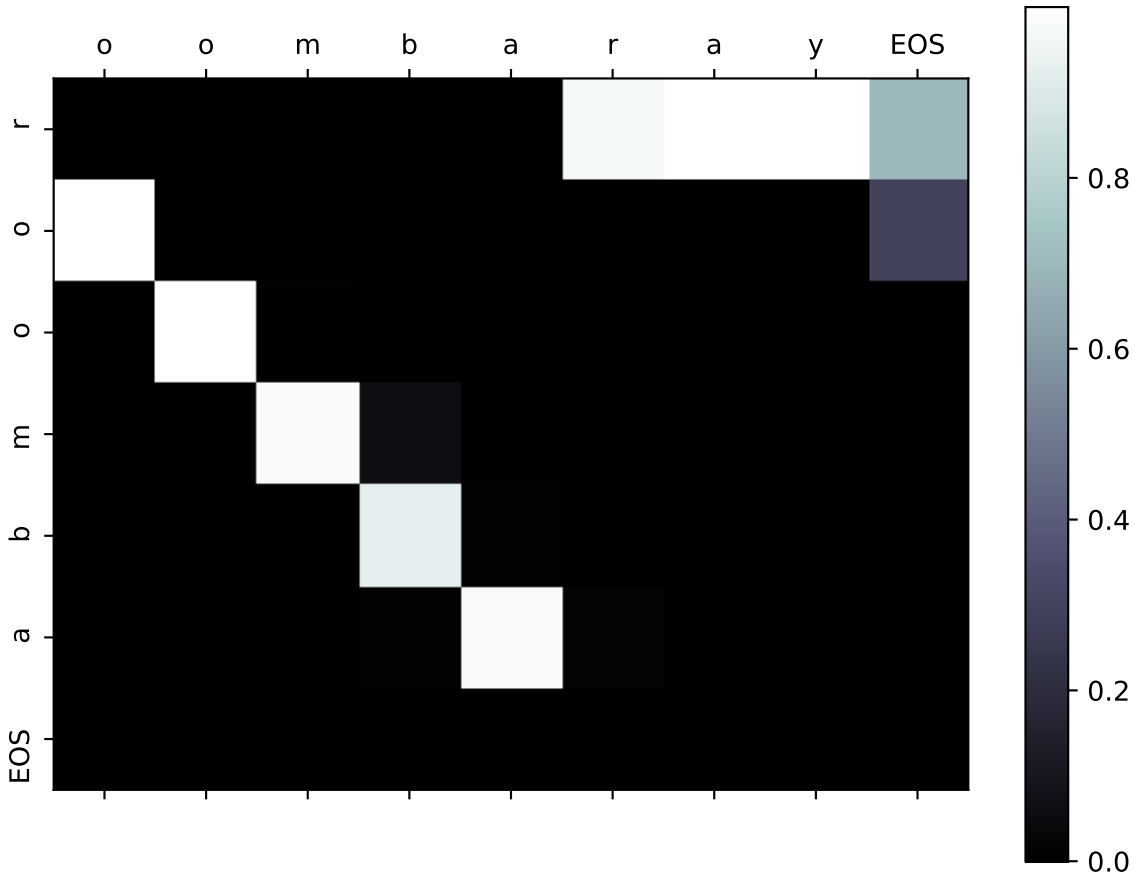

Fig 1.3 roomba

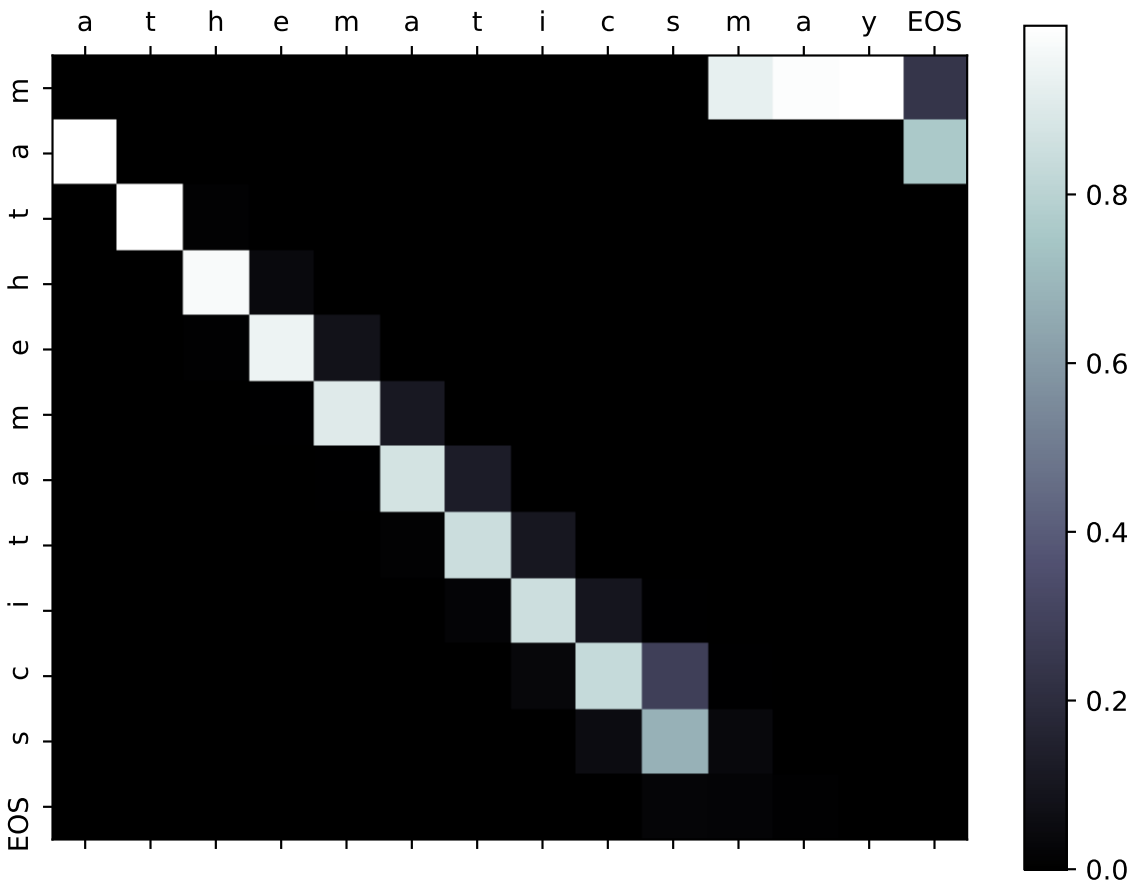

Fig 1.4 mathematics