Part1

1. I think it will perform bad on long sequence. Since if the sentence is too long, we might lose some information from the previous part, and it is easy to make mistake.

2. We can use gradient clipping or LSTM to improve performance.

3. Science every time we are inputting the output from the previous step, then if one output is wrong, the rest of the term might wrong.

4. To solve this we can pass every information we need from the previous step to next step instead of truth token.

Part3

2. I think the result performs really bad and unstable for any type of word.

3. I tried few sentences. For example, for 'my name is yiming zhang', the result is 'yway amengway issay yinghay-inway-awlay anghay'. It is easy to figure out that there are many failures such as 'name' to 'amengway' and so on.

Part4

3. The result is better than 'RNN decoder' for every type of words, and rarely see failure mode. The speed is slower science the number of parameters that needs to be calculates are increased.

4. The result is worse, and speed is faster. The result is worse is because for long or complex words it is hard to deal with by only dot product. The speed is faster is because for dot product, it is faster to compute.

Part5

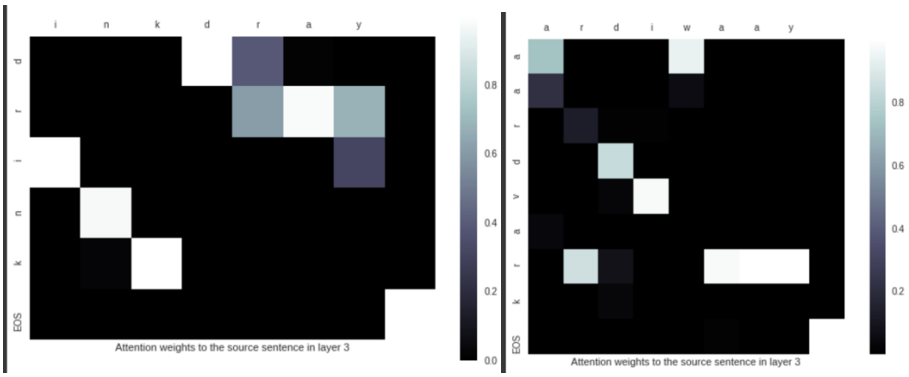
1. As I explained in the previous question, the speed of the scaled dot-product is faster and additive attention is slow, but the result of additive attention is better than scaled dot-product.

3. The result is better, and the speed is faster.

4. The result is worse, and the speed is similar.

5.

Part6



drink aardvark

As we can see from the result, short words are performed well and more likely to produce correct result. As I showed for word ‘drink’, it produced ‘inkdray’.

On the other hand, while handling long and unusual words it is easier to make mistake. The second picture shows that ‘aardvark’ are translated to ‘ardiwaay’. This might because the network is not trained enough with those complex cases, so it performed poorly.