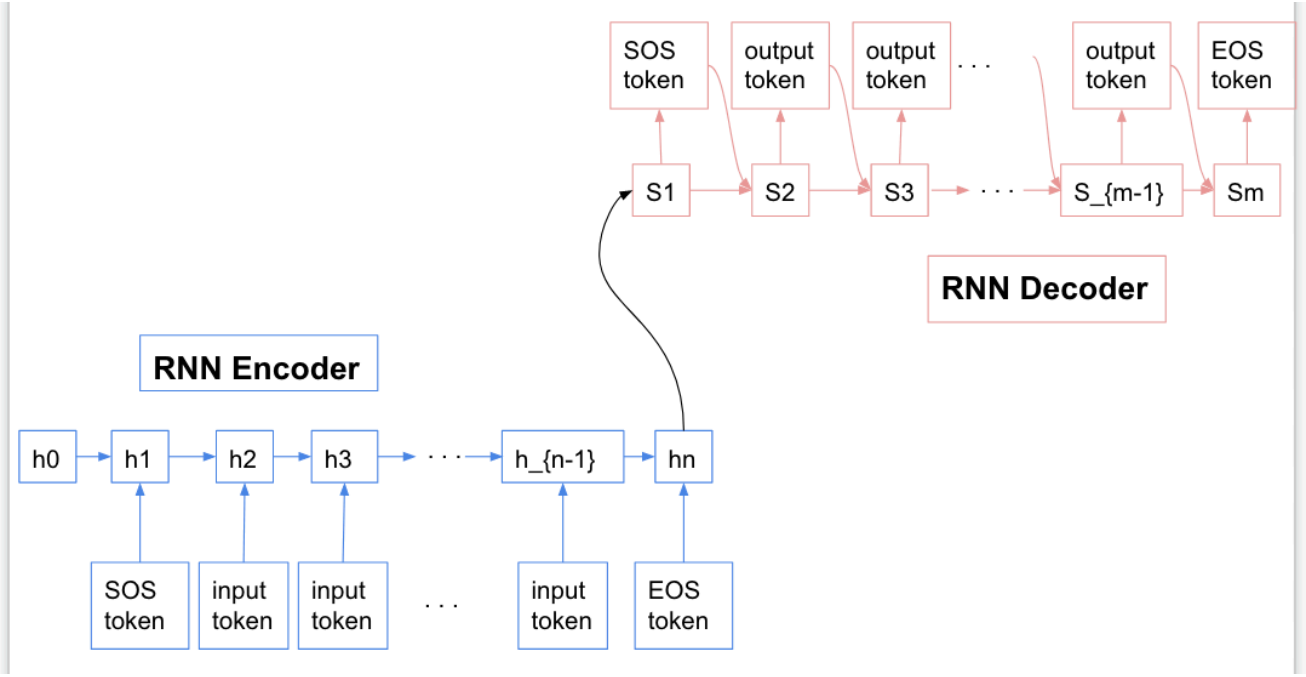
HW4: Seq2Seq Modeling

Jing Qian (jq2282)

2.1. Model architecture



The seq2seq model is consisted of two main parts: RNN encoder and RNN decoder. For the encoder part, the input is a sequence of input tokens starting with the start-of-sequence (SOS) token and ending with the end-of-sequence (EOS) token. h_n refers to the final hidden state of the encoder. For the decoder part, the decoder input is initialized as the numerical identifier of SOS token and the decoder hidden state is initialized as h_n. The RNN decoder outputs a sequence of tokens, starting with SOS token and possibly ending with EOS tokens. If the output reaches the length limit, the final token may not be EOS token.

2.2. Error analysis

Here I use the greedy decoding on the developement dataset to do the error analysis. Three kinds of erros I found are:

1. The model mistakes two similar geographical attributes: "near[]" and "area[]". For example, the input includes 'area[city centre]' and 'near[The Sorrento]', and the prediction is "The mill is a coffee shop in in the sorrento..." which should be "in the city centre area near the Sorrento" instead.
2. The model learnt the relationship between values of different attributes instead of the

relationship between attribute and corresponding values. For example, the input includes 'name[The Wrestlers]', 'eatType[coffee shop]' and 'food[Chinese]' and the prediction is "The wrestlers is a coffee shop providing Indian food..." which should be "providing Chinese food". In fact, the training dataset contains examples of "The Wrestlers" providing "Chinese food". So the model picked "Chinese food" instead of "Indian food".
3. Repeated information. One example is: the input contains 'area[Portland Arms]' and the prediction is "Cotto is a coffee shop in the Portland Arms in Portland Arms". The greedy decoder outputs the word with highest probability and hence may not show the information correctly.

2.3. Beam search analysis

2.4. What's wrong with BLEU?

The problems with BLEU scores are:

1. They tend to favor short translations even with brevity penalty.
2. The predictions may not be fluent or correct in grammar or semantics.
3. The performance may be over-estimated by the appearance of common words.
4. The performance may be under-estimated because BLEU fails to recognize the synonyms or paraphrases.

So BLEU may over- or under-estimate performance. In our case, since we use model generation, it tends to over-estimate performance more. Here are examples:

1. The input is ('name[Fitzbillies]', 'eatType[coffee shop]', 'food[English]', 'priceRange[cheap]', 'customer rating[5 out of 5]', 'area[riverside]', 'familyFriendly[no]') and the prediction "Fitzbillies coffee shop in Riverside. It is not a coffee shop with a customer rating and is not family-friendly." got a BLEU score 0.49. It misused the "no" in "familyFriendly" in the "eatType" so the meaning is distorted. But BLEU could not figure it out.
2. The input is ('name[Cocum]', 'eatType[coffee shop]', 'food[Chinese]', 'priceRange[less than £20]', 'customer rating[low]', 'familyFriendly[yes]') and the prediction "Cocum is a coffee shop serving Indian food and is less than £20." got a BLEU score 0.44. The prediction made mistakes and lost almost half of the information in the input due to the short length. But since most words in the prediction also appeared in the reference sentences, the BLEU score over-estimate the performance.

The Benefits ot BLEU are: it's automated, quick and inexpensive. It is reported to be correlated well with human judgement. Although it has multiple problems, it is still a widely used metrics.

2.5. Pyramid scoring for diverse outputs.

1. Two SCUs (weight = 3) are:

- The Cambridge Blue is family friendly,
- The Cambridge Blue is a coffee shop.

2. Two SCUs (weight = 2) are:

- The Cambridge Blue is in the riverside area.
- The Cambridge Blue is near Burger King.

3. Two SCUs (weight = 1) are:

- The Cambridge Blue is low-priced.
- The Cambridge Blue provides Italian food.

The Pyramid score is D/Max, which is the the sum of the weights of the SCUs in a summary divided by the sum of the weights of the SCUs in a ideally informative summary. The generated sentence has 3 SCUs with weight = 3 : "The Cambridge Blue", "family-friendly" and "coffee shop", 2 SCUs with weight = 2: "near Burger King" and "at the riverside", so the D = 3*3 + 2*2 = 13. Other than the SCUs in the generated sentence, the ideally informative summary would also include 1 SCU with weight = 3: "average customer rating" and 2 SCUs with weight = 1: "low-priced" and "Italian food". So Max = 3*4+2*2 + 1*2 = 18. So the Pyramid score for the generated sentence is 13/18.

2.6. How could we deal with unseen restaurant names?

I tested the unseen restaurant names as "Jings Malatang". The entire inputs are: ('name[Jings Malatang]','eatType[coffee shop]', 'priceRange[high]', 'area[city centre]', 'near[The Sorrento]'). The greedy decoder prediction is "The Mill is a coffee shop near The in in the city centre near The is" while the beam-search decoder predictions are "The Sorrento in the city centre which is a coffee shop.", "The Sorrento in the city centre which is a coffee shop that is a coffee shop with a high customer rating." and "The Sorrento in the city centre which is a coffee shop that is a coffee shop". From the result we could see, the model could not identify the unseen restaurant names as names, so it either took the common restaurant names in its training set or took the restaurant names from the "near" attribute.

So the issue here is how we build the connection between attribute "name" with its value in the brackets. We could add attention to the encoder. Or, we could do some mapping between the attribute and its values, which means we don't train the restaurant names themselves, we just use some symbols to represent this entity and replace the symbols to actual names after decoder output.