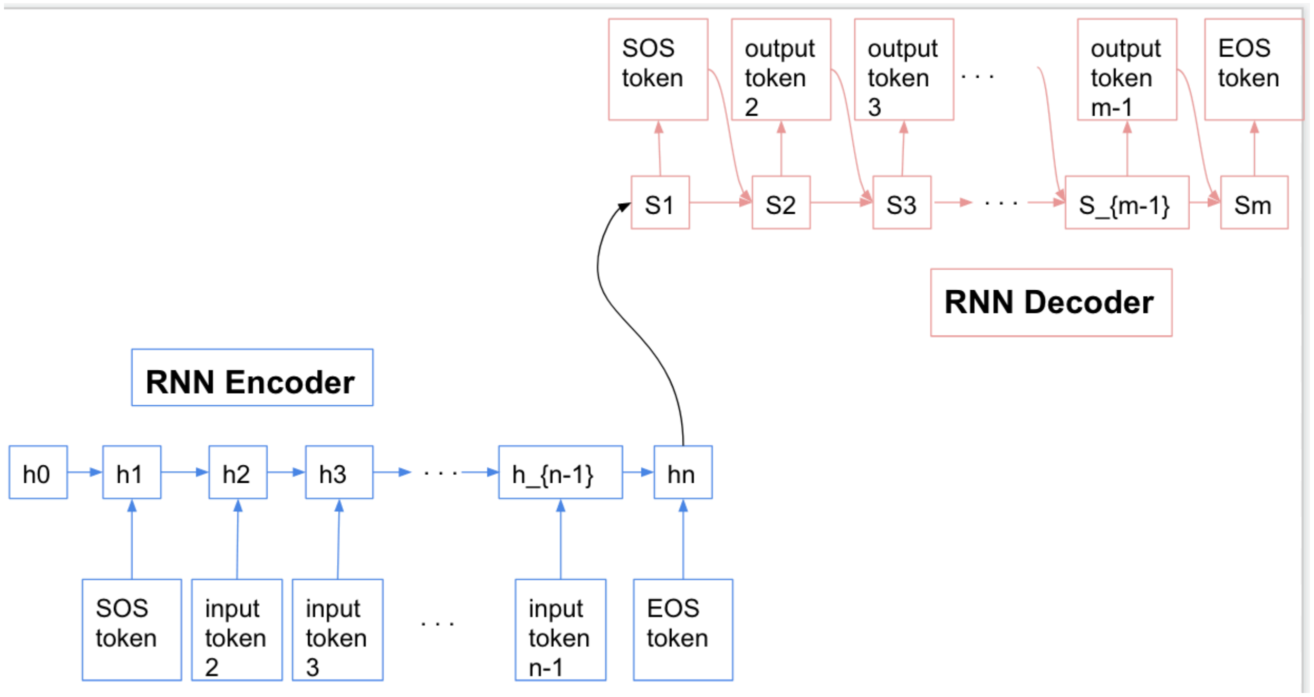
HW4: Seq2Seq Modeling

Jing Qian (jq2282)

Dear instructors, I want to use **2 late days** for this homework.

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. Here input tokens (2 to n-1, corresponding to the name of h) are tokens from short text. 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. Here the output tokens are tokens from words (2 to m-1, corresponding to the name of S). 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

Example 1:

Input sequence: ('name[The Mill]', 'eatType[coffee shop]', 'food[Chinese]', 'priceRange[high]', 'area[city centre]', 'near[The Sorrento]')

Greedy decoder prediction: " The Mill is a coffee shop located near The located in the The Sorrento. " with BLEU score: 0.45.

Beam search decoder prediction:

" The Mill is an Italian coffee shop located near The Sorrento in the city centre. with BLEU score 0.51.

" The Mill is an Italian coffee shop located near The Sorrento. It is located in the city centre. " with BLEU score 0.53.

" The Mill is an Italian coffee shop located near The Sorrento. It is located in the riverside. " with BLEU score 0.41.

Analysis: Here beam search doesn't include the greedily decoder output. The three outputs have the same start of the sentence but later become different. We could see that the beam search outputs have better grammar performance than greedy output. All three beam search outputs contain more information than the greedy one. However, some of the information didn't appear in the input, which may exist in the training data. We could see that the second output has more words than the first one, although they both contains the same information. Due to the different length, the second output has higher BLEU score. The second and third output has similar lengths, but due to the wrong information in the third output, it even has lower BLEU score than greedy.

Example 2:

Input sequence: ('name[The Golden Palace]', 'eatType[coffee shop]', 'food[English]', 'priceRange[£20-25]', 'customer rating[high]', 'area[city centre]')

Greedy decoder prediction: " The Golden Palace is a coffee shop in the city city centre. It a city centre. It is It is is and is " with BLEU score: 0.25

Beam search decoder prediction:

" In the city centre, The Golden Palace is a highly rated coffee shop that serves English food. " with BLEU score 0.49.

" In the city centre, The Golden Palace is a highly rated coffee shop that serves English food. It is located in the city centre. " with BLEU score 0.53.

" In the city centre, The Golden Palace is a highly rated coffee shop that serves English food. It is located in the city centre and is rated highly by customers. " with BLEU score 0.4.

Analysis: Here beam search doesn't include the greedily decoder output. The sentence lengths vary among the three outputs of beam search, but they contain similar information, more than that in greedily decoder output. The three outputs have the same start of the sentence but later become different. The different lengths due to the repeated information in the outputs, which leads to different BLEU scores.

Example 3:

Input sequence: ('name[Fitzbillies]', 'eatType[coffee shop]', 'food[Chinese]', 'priceRange[less than £20]', 'customer rating[low]', 'area[riverside]', 'familyFriendly[no]')

Greedy decoder prediction: " Fitzbillies is a low priced family friendly coffee shop located in the city centre. " with BLEU score: 0.35.

Beam search decoder prediction:

" Fitzbillies is a low priced family friendly coffee shop serving Chinese food. It is located in the city centre. with BLEU score 0.52.

" Fitzbillies is a low priced family friendly coffee shop serving Chinese food. It is located in the city centre and has a low customer rating. " with BLEU score 0.5.

" Fitzbillies is a low priced family friendly coffee shop located in the city centre. " with BLEU score 0.35.

Analysis: Here the beam search includes the greedily decoded output, as the third one. The three outputs have the same start of the sentence but later become different. The first two outputs are similar in length, information and BLEU scores, which are much better than the greedy one. One weird thing is although output 2 contains more correct information than the first one, it has lower BLEU score. This may due to the paraphrase issue in the reference sentences.

No, beam search doesn't always include the greedily decoded output because I added the length penalty in the sequence likelihood. Hence the most likely word in greedy decoding may not beat other less likely words in the beam. As shown in the examples, only beam search in example 3 includes the greedily decoded output.

Advantages of beam search: at each decoding step, beam search decoding cosses the top k most likely sequences instead of the most likely sequence in the greedy decoding and hence it has a broader search space than greedy and provides multiple output finally. Also, the beam size is a tunable parameter, which could be used to control the search process and get better result.

Disadvantanges of beam search: the beam search expands more space than the greedy decoding and hence it takes longer running time and larger space. Moreover, since it keeps beam-size number of best partial solutions as candidates, the output could not be guaranteed to be optimal.

From the three examples, we could see that the BLEU score of three beam search outputs vary: some are better than greedy and some are worse. Generally speaking, beam search has a higher BLEU score than greedy.

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 . 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:

- It is family friendly,

- It is a coffee shop.

2. Two SCUs (weight = 2) are:

- It is in the riverside area.
- It is near Burger King.

3. Two SCUs (weight = 1) are:

- It is low-priced.
- It 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. Here "ideally informative summary" could be understood in multiple ways and I consider it as a summary that includes all SCUs in all reference summaries.

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."

"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.

2.7. What problems might occur in different languages?

I think E2E challenge might be harder in Chinese than in English. One big difference between English and Chinese is that: the unit of English is words, which are segmented by blank space while the unit of Chinese is characters, which don't have a natural segmentation. So when doing Chinese NLP task, one extra step than English NLP task is the segmentation of characters, which turn a group of characters into words and different character segmentations may lead to different words (some segmentation may not be meaningful). This may lower the performance of the seq2seq model, in the decoder part, for example.