cs224n

(arthurg)

May 27, 2020

Question 1.

The masks is 1 where the padding occurs, 0 elsewhere. In the attention function, we give an attention score of -infinity where the padding occured. Then, after we apply softwmax function, whever the padding occurs will always have a very low probability close to 0.

This is necessary because we don't want to give attention to padding tokens during the decoding process (as their corresponding encoder states contain no real information).

Question 2.

My model's BLEU score is 35.69179114233418

Question 3.

The NMT translated mistook "one of my favourites" for "favourite of my favourites."

In the third decoding step (when trying to decode "one"), the attention was improperly focused on "favoritos" instead of "otro". To fix, we need better attention scores. This could be done by stacking more layers in our attention mechanism (currently we have $W_{attProj} \odot h_i^{enc}$ which is technically only 1 linear layer). In addition, we might also be able to fix it by getting more training data.

Question 4.

The NMT outputted "the author for children, more reading in the U.S." instead of "Americas most widely read childrens author, in fact."

The issue is that the words are mostly accurate, however the order and the grammar seems slightly off. To fix this issue, i would incorporate a pre-trained language model into the model, similar to what is done in SMT.

Question 5.

The NMT system outputted "Richard <unk>" instead of "Richard Bolingbroke"

The current NMT system is not programmed to handle OOV words. When ever the decoder is expecting to output an OOV word, we could instead output the corresponding input word with the highest attention score. Many OOV words are things that like proper nouns (eg: names), which don't change between different languages.

Question 6.

The NMT outputted "back to the apple" instead of "go around the block"

The issue is that in spanish "manzana" can either translate to "apple" or "block". We can also fix this with a language model as suggested 2 questions ago. We can also try training the NMT on less formal text (eg: casual conversations) to try to pick up figures of speech such as "around the block"

Question 7.

The NMT outputted "womens room" instead of "teachers lounge."

There could be multiple issues that could be causing this translation. One explanation is that there is bias in our word embeddings, where "women" and "teacher" were associated closely (as discussed in A1). Another explanation is that the attention mechanism is not working properly, and the attention score for "Elle" and "al bano" were too high when trying to decode "teachers".

The fix for the first issue is to get better pre-trained embeddings. The fix for the second issue is to get better attention mechanism as previously discussed.

Question 8.

The NMT outputted "250 thousand acres" instead of "100,000 acres"

The main issue is that the NMT probably has not seen enough example of translating roman numbers in the training data. Depending on how the numerical "words" are structured, we could be translating each arabic numeral as a word. This means that when we use our attention mechanism has to be know how to handle every single power of 10 (eg: diff attention mechanism needs to know about 1-9, 10-99, 100-999, 1000-9999).

To handle this better, we could probably write a rule keep roman numbers as OOV and translate them by copying the input directly to the output if the attention is high on the number (similar to what is proposed for proper nouns).

Question 9.

Source Sentence: Puedo vestirme como agricultor, o con ropa de cuero, y nunca nadie ha elegido un agricultor.

NMT Output: I can use as a farmer , or leather leather , and no one has ever chosen a farmer .

Reference Output: You can have me as a farmer, or in leathers, and no one has ever chose farmer.

Error in NMT Translation: There are a few errors in the translation. NMT output says "I can use as a farmer", whereas reference says "You can have me as a farmer." This example shows that the tranlation wasn't able to find out which where to place the subject ("you") and where to place the object ("me"). In the translation, the NMT had used "me" twice, and did not properly use the object.

Possible Fix: To fix this issue, we can either hardcode some of these more complex grammar rules, or we can try increasing the beam size during decoding. This problem seems like it was a result of being too greedy while decoding.

Source Sentence: podras venirte a Ohio, y por favor trae a ese hombre que s que ya has encontrado.

NMT Output: You could put it to Ohio , and please bring that man who I know you 've already found .

Reference Output: Could you please come to Ohio, and please bring that man that I know you have found by now.

Error in NMT translation: NMT says "You could put it to Ohio" but the reference says "Could you please come to Ohio." The NMT system got the main idea right, but used "put it" instead of "come to." The NMT system didn't realize that "put" technically agrees with object "it", but the correct object shoudl be "you" (and thus "come" is a better verb than "put"). To make sure that the objects are correct, I'd recommend that we make our decoder more complicated. This could be by adding a more complex LM, adding more layers, or using a bigger beam.

Question 10.

Candidate Translation 1: the love can always do

unigram: the, love, can, always do

Unigram	Count C	Count R	min (countC, countR)
the	1	0	0
love	1	1	1
can	1	1	1
always	1	1	1
do	1	0	0

$$p_1 = \frac{\sum min(countC, countR)}{\sum countC}$$
$$p_1 = \frac{0+1+1+1+0}{1+1+1+1+1}$$
$$p_1 = \frac{3}{5}$$

bigram: the love, love can, can always, always do

Bigram	Count C	Count R	min (countC, countR)
the love	1	0	0
love can	1	1	1
can always	1	1	1
always do	1	0	0

$$p_2 = \frac{\sum min(countC, countR)}{\sum countC}$$
$$p_2 = \frac{0+1+1+0}{1+1+1+1}$$
$$p_2 = \frac{2}{4} = \frac{1}{2}$$

$$len(c1) = 5$$
$$len(r1) = 6$$
$$len(r2) = 4$$

So the BP is $exp(1 - \frac{4}{5}) = exp(\frac{1}{5})$

$$BLEU_1 = BP * exp(\lambda_1 * logp_1 + \lambda_2 * logp_2)$$
$$BLEU_1 = exp(\frac{1}{5}) * exp(0.5 * log\frac{3}{5} + 0.5 * log\frac{1}{2})$$
$$BLEU_1 = 0.66899$$

So the BLEU score is 0.66899 for candidate translation 1.

Candidate Translation 2: love can make anything possible unigram: love, can, make, anything, possible

Unigram	Count C	Count R	min (countC, countR)
love	1	1	1
can	1	1	1
make	1	1	1
anything	1	1	1
possible	1	1	1

$$p_1 = \frac{\sum min(countC, countR)}{\sum countC}$$
$$p_1 = \frac{1+1+1+1+1}{1+1+1+1+1}$$
$$p_1 = \frac{5}{5} = 1$$

bigram: love can, can make, make anything, anything possible

Bigram	Count C	Count R	min (countC, countR)
love can	1	1	1
can make	1	1	1
make anything	1	1	1
anything possible	1	1	1

$$p_2 = \frac{\sum min(countC, countR)}{\sum countC}$$
$$p_2 = \frac{1+1+1+1}{1+1+1+1}$$
$$p_2 = \frac{4}{4} = 1$$

$$len(c2) = 4$$
$$len(r1) = 6$$
$$len(r2) = 4$$

So the BP is 1

$$BLEU_2 = BP * exp(\lambda_1 * logp_1 + \lambda_2 * logp_2)$$
$$BLEU_2 = 1 * exp(0.5 * log1 + 0.5 * log1)$$
$$BLEU_2 = 0$$

So the BLEU score is 0 for candidate translation 2.

The BLEU score says translation 2 is better. I agree because it is spot on to the reference.

Question 11.

Candidate Translation 1: the love can always do

unigram: the, love, can, always do

Unigram	Count C	Count R	min (countC, countR)
the	1	0	0
love	1	1	1
can	1	1	1
always	1	1	1
do	1	0	0

$$p_1 = \frac{\sum min(countC, countR)}{\sum countC}$$
$$p_1 = \frac{0+1+1+1+0}{1+1+1+1+1}$$
$$p_1 = \frac{3}{5}$$

bigram: the love, love can, can always, always do

Bigram	Count C	Count R	min (countC, countR)
the love	1	0	0
love can	1	1	1
can always	1	1	1
always do	1	0	0

$$p_2 = \frac{\sum min(countC, countR)}{\sum countC}$$
$$p_2 = \frac{0+1+1+0}{1+1+1+1}$$
$$p_2 = \frac{2}{4} = \frac{1}{2}$$

$$len(c1) = 5$$
$$len(r1) = 6$$

So the BP is $exp(1 - \frac{6}{5}) = exp(-\frac{1}{5})$

$$BLEU_1 = BP * exp(\lambda_1 * log p_1 + \lambda_2 * log p_2)$$
$$BLEU_1 = exp(-\frac{1}{5}) * exp(0.5 * log\frac{3}{5} + 0.5 * log\frac{1}{2})$$
$$BLEU_1 = 0.448437$$

So the BLEU score is 0.448437 for candidate translation 1.

Candidate Translation 2: love can make anything possible unigram: love, can, make, anything, possible

Unigram	Count C	Count R	min (countC, countR)
love	1	1	1
can	1	1	1
make	1	0	0
anything	1	0	0
possible	1	0	0

$$p_1 = \frac{\sum min(countC, countR)}{\sum countC}$$
$$p_1 = \frac{1+1+0+0+0}{1+1+1+1+1}$$
$$p_1 = \frac{2}{5}$$

bigram: love can, can make, make anything, anything possible

Bigram	Count C	Count R	min (countC, countR)
love can	1	1	1
can make	1	0	0
make anything	1	0	0
anything possible	1	0	0

$$p_2 = \frac{\sum min(countC, countR)}{\sum countC}$$
$$p_2 = \frac{1+0+0+0}{1+1+1+1}$$
$$p_2 = \frac{1}{4}$$

$$len(c2) = 4$$
$$len(r1) = 6$$

So the BP is $exp(1 - \frac{4}{6}) = exp(\frac{1}{3})$

$$BLEU_2 = BP * exp(\lambda_1 * logp_1 + \lambda_2 * logp_2)$$
$$BLEU_2 = exp(\frac{1}{3}) * exp(0.5 * log\frac{2}{5} + 0.5 * log\frac{1}{4})$$

$$BLEU_2 = exp(\frac{1}{3}) * exp(0.5 * log\frac{2}{5} + 0.5 * log\frac{1}{4})$$

So the BLEU score is 0.448437 for candidate translation 2.

The BLEU score says translation 1 is better. I don't agree, because translation 2 was better able to convey the intended meaning.

Question 12.

it is problematic when using only a single reference translation because often times in english, you can phrase the same translation in multiple ways. If the NMT phrases something correctly (but not the same as the reference), it should still have a high BLEU score.

Question 13.

Advantage

- Very objective, no arbitrariness of human judges

- Very fast and cheap to compute

Disadvantage

- BLEU knows nothing about the language, and checks for the presnse of key words and phrases instead of proper grammer rules

- BLEU may give arbitrarily low scores if NMT phrases something differently but correctly