Bidirectional Decoding for seq2seq

Su Jianlin

August 9, 2019

In the article "Playing with Keras: seq2seq Automatic Title Generation", we basically explored seq2seq and provided a reference Keras implementation.

This article takes that seq2seq a step further by introducing a bidirectional decoding mechanism. To a certain extent, it can improve the quality of generated text (especially when generating longer texts). The bidirectional decoding mechanism introduced in this article refers to "Synchronous Bidirectional Neural Machine Translation", and I have implemented it using Keras.

1 Background Introduction

Readers who have studied seq2seq know that the common seq2seq decoding process is generated word-by-word (or token-by-token) from left to right. That is, the first word is generated based on the encoder's results; then the second word is generated based on the encoder's results and the already generated first word; then the third word is generated based on the encoder's results and the first two words; and so on. In general, it models the following probability decomposition:

$$p(Y|X) = p(y_1|X)p(y_2|X, y_1)p(y_3|X, y_1, y_2)\cdots \tag{1}$$

Of course, it can also be generated from right to left, i.e., generating the last word first, then the second-to-last word, the third-to-last word, and so on. The problem is that no matter which direction is used for generation, there will be a problem of directional bias. For example, if generating from left to right, the generation accuracy of the first few words will definitely be higher than that of the last few words, and vice versa. In "Synchronous Bidirectional Neural Machine Translation", the following statistical results on machine translation tasks are given:

Model	The first 4 tokens	The last 4 tokens
L2R	40.21%	35.10%
R2L	35.67%	39.47%

Table 1: Accuracy comparison between Left-to-Right (L2R) and Right-to-Left (R2L) decoding.

L2R and R2L refer to left-to-right and right-to-left decoding generation, respectively. From the table, we can see that if decoding from left to right, the accuracy of the first four tokens is about 40%, but the accuracy of the last four tokens is only 35%; the reverse is also true. This reflects the asymmetry of decoding.

To eliminate this asymmetry, "Synchronous Bidirectional Neural Machine Translation" proposes a bidirectional decoding mechanism that maintains two directional decoders and then uses Attention to further align the generation.

2 Bidirectional Decoding

Although this article refers to "Synchronous Bidirectional Neural Machine Translation", I have not fully read the original text in detail. I implemented the model based on my own intuition after roughly reading the original text and understanding the principles, so I do not guarantee that it is exactly the same as the original text. Furthermore, this paper is not the first to work on bidirectional decoding generation, but it is the first paper I saw on bidirectional decoding, so I only implemented it and did not compare it with other related papers.

2.1 Basic Idea

Since it is called bidirectional "decoding", the changes are only in the decoder and do not involve the encoder. Therefore, the following introduction only focuses on describing the decoder part. Also, it should be noted that bidirectional decoding is just a strategy, and the following is only a reference implementation, not the standard or unique one. This is just like how we say seq2seq is just a general term for sequence-to-sequence generation models; there are many adjustable parts in how the specific encoder and decoder are designed.

First, here is a simple descriptive placeholder for the interaction process (the original post included an animation):

Click here to view the Video: Seq2Seq Bidirectional Decoding Mechanism Animation

Figure 1: Illustration of the Bidirectional Decoding Mechanism for Seq2Seq

As shown in the figure, bidirectional decoding can basically be seen as the coexistence of two decoding modules in different directions. For convenience of description, we call the upper part the L2R module and the lower part the R2L module. At the start, everyone inputs a start marker (S in the figure), then the L2R module is responsible for predicting the first word, and the R2L module is responsible for predicting the last word. Next, the first word (and historical information) is passed into the L2R module to predict the second word. To predict the second word, in addition to using the L2R module's own encoding, the existing encoding results of the R2L module are also used. Conversely, the last word (and historical information) is passed into the R2L module, plus the existing encoding information of the L2R modul to predict the second-to-last word; and so on, until the end marker (E in the figure) appears.

2.2 Mathematical Description

In other words, **when each module predicts each word, in addition to using the information inside the module, it also uses the encoded information sequence of the other module**, and this "use" is achieved through Attention. Using formulas, suppose that in the current situation, the L2R module needs to predict the n-th word, and the R2L module needs to predict the n-th word from the end. Suppose that after several layers of encoding, the obtained L2R vector sequence is:

$$H^{(l2r)} = \left[h_1^{(l2r)}, h_2^{(l2r)}, \dots, h_n^{(l2r)}\right] \tag{2}$$

And the R2L vector sequence is:

$$H^{(r2l)} = \left[h_1^{(r2l)}, h_2^{(r2l)}, \dots, h_n^{(r2l)}\right] \tag{3}$$

If it were unidirectional decoding, we would use $h_n^{(l2r)}$ as a feature to predict the n-th word, or use $h_n^{(r2l)}$ as a feature to predict the n-th word from the end.

In the bidirectional decoding mechanism, we use $h_n^{(l2r)}$ as the query, and then use $H^{(r2l)}$ as the key and value to perform an Attention. The output of the Attention is used as a feature to predict the n-th word. In this way, when predicting the n-th word, the subsequent words can be "perceived" in advance. Similarly, we use $h_n^{(r2l)}$ as the query, and then use $H^{(l2r)}$ as the key and value to perform an Attention. The output of the Attention is used as a feature to predict the n-th word from the end. In this way, when predicting the n-th word from the end, the preceding words can be "perceived" in advance. In the schematic diagram above, the interaction between the top two layers and the bottom two layers refers to Attention. In the code below, the most common multiplicative Attention is used (refer to "A Shallow Reading of 'Attention is All You Need' (Introduction + Code)").

3 Model Implementation

The above is the basic principle and practice of bidirectional decoding. It can be felt that in this way, the seq2seq decoder also becomes symmetrical, which is a very beautiful feature. Of course, to fully implement this model, some questions need to be considered: 1. How to train? 2. How to predict?

3.1 Training Scheme

Like ordinary seq2seq, the basic training scheme is to use the so-called Teacher-Forcing method for training. That is, when the L2R direction predicts the n-th word, it is assumed that the first $n-1$ words are accurately known, and when the R2L direction predicts the n-th word from the end, it is assumed that the $n-1, n-2, \ldots, 1$ words from the end are accurately known. The final loss is the average of the word-by-word cross-entropy of the two directions.

However, such a training scheme is a last resort, and we will analyze the drawbacks of its information leakage later.

3.2 Bidirectional Beam Search

Now let's discuss the prediction process.

If it is a conventional unidirectional decoding seq2seq, we would use the beam search algorithm to give a sequence with as high a probability as possible. Beam search refers to decoding word by word in sequence, keeping only the top-k "temporary paths" with the highest probability each time until the end marker appears.

In bidirectional decoding, the situation becomes a bit more complicated. We still use the idea of beam search, but simultaneously cache the top-k results of both directions. That is to say, the L2R and R2L directions each store top-k temporary paths. In addition, since the L2R decoding refers to the existing R2L decoding results, when we want to predict the next word, in addition to enumerating the top-k words with the highest probability and enumerating the top-k L2R temporary paths, we also have to enumerate the top-k R2L temporary paths. Therefore, a total of $topk^3$ combinations must be calculated. After the calculation is completed, a simple idea is adopted: the score of each "word - L2R temporary path" is averaged over the "R2L temporary path" dimension, so that the score returns to $topk^2$ items. These are used as the score for each "word - L2R temporary path", and then the top-k with the highest scores are selected from these $topk^2$ combinations. The decoding on the R2L side undergoes the same processing in reverse. Finally, if both the L2R and R2L directions have decoded complete sentences, the one with the highest probability (score) is selected.

This entire process is called "Bidirectional Beam Search". If readers are familiar with unidirectional beam search or have even written beam search themselves, the above process is actually not difficult to understand (it is easier to understand by looking at the code); it can be

considered a natural extension of unidirectional beam search. Of course, if you don't understand beam search itself, the description of the above search process might be confusing. Therefore, readers who want to understand the principle should start from the conventional unidirectional beam search, understand it first, then look at the description of the above decoding process, and finally look at the reference code provided below.

3.3 Code Reference

Below is the reference implementation of bidirectional decoding provided by the author. The overall structure is consistent with the previous "Playing with Keras: seq2seq Automatic Title Generation", but the decoding end has been changed from unidirectional to bidirectional:

https://github.com/bojone/seq2seq/blob/master/seq2seq_bidecoder.py

Note: The test environment is still similar to before, roughly Python 2.7 + Keras 2.2.4 + Tensorflow 1.8. For friends using Python 3.x or other environments, if you can modify it yourself, please make the corresponding changes. If you cannot modify it yourself, please do not ask me. I really don't have the time or obligation to help you run it in every environment. Can we just discuss seq2seq technology-related content in this article?

In this implementation, I think it is necessary to explain the start and end markers. In the previous unidirectional decoding example, I used 2 as the start marker and 3 as the end marker. In bidirectional decoding, a natural question is: Should the L2R and R2L directions use two sets of start and end markers?

Actually, there should be no standard answer. I think whether sharing one set or maintaining two sets of start and end markers, the results might be similar. As for the scheme used in the above reference code, it is a bit unconventional, but I think it is more intuitive. Specifically: Only one set is still used, but in the L2R direction, 2 is used as the start marker and 3 as the end marker, wh

3.4 Thinking and Analysis

Finally, let's think further about this bidirectional decoding scheme. Although symmetrizing the decoding process is a very beautiful feature, it doesn't mean it's completely without problems. Thinking about it more deeply helps us better understand and use it.

1. Reasons for improved generation

An interesting question is: It seems that bidirectional decoding can indeed improve the generation quality of the beginning and end of the sentence, but will it simultaneously reduce the generation quality of the middle part?

Of course, theoretically this is possible, but it is not very serious in actual testing. On the one hand, the information encoding and decoding capabilities of the seq2seq architecture are still very strong, so information will not be easily lost. On the other hand, when we evaluate the quality of a sentence ourselves, we often focus on the beginning and end parts. If the beginning and end parts are reasonable and the middle part is not too bad, then we consider it a reasonable sentence. Conversely, if the beginning or end is unreasonable, we feel the sentence is very bad. In this way, by improving the generation quality of the beginning and end of the sentence, the overall generation quality is improved.

2. Mismatch with probability models

For unidirectional decoding, we have a clear probabilistic explanation, i.e., estimating the conditional probability $p(Y|X)$ (which is Eq. 1). But in bidirectional decoding, we find that we don't know how to map it to a probability model at all. In other words, we feel we are calculating

Model	DEV	MT03	MT04	M05	MT06	AVE	Δ
Moses	37.85	37.47	41.20	36.41	36.03	37.78	-9.41
RNMT	42.43	42.43	44.56	41.94	40.95	42.47	-4.72
Transformer	48.12	47.63	48.32	47.51	45.31	47.19	-
Transformer (R2L)	47.81	46.79	47.01	46.50	44.13	46.11	-1.08
Rerank-NMT	49.18	48.23	48.91	48.73	46.51	48.10	+0.91
ABD-NMT	48.28	49.47	48.01	48.19	47.09	48.19	+1.00
Our Model	**50.99**	**51.87**	**51.50**	**51.23**	**49.83**	**51.11**	**+3.92**

Table 3: Evaluation of translation quality for Chinese-English translation tasks using case-insensitive BLEU scores. All results of our model are significantly better than Transformer and Transformer (R2L) ($p < 0.01$).

Figure 2: Improvement of bidirectional decoding relative to other unidirectional models in the original paper.

probabilities and the effect is there, but we don't know what we are truly calculating because the conditional dependence of the conditional probability has been completely disrupted.

Of course, if it is truly effective, it doesn't matter if the theoretical beauty is a bit lacking. This point I mentioned is just a pursuit of theoretical aesthetics; everyone can have their own opinion.

3. Information leakage

The so-called information leakage refers to the fact that the labels that were originally intended to be prediction targets are used as inputs, resulting in a falsely low loss (or falsely high accuracy) during the training phase.

Since in bidirectional decoding, the decoding of the L2R end needs to read the existing vector sequence of the R2L end, and in the training phase, to predict the n-th word of the R2L end, the first $n-1$ words need to be passed in. In this way, the further the decoding goes, the more serious the information leakage becomes. As shown in the figure below:

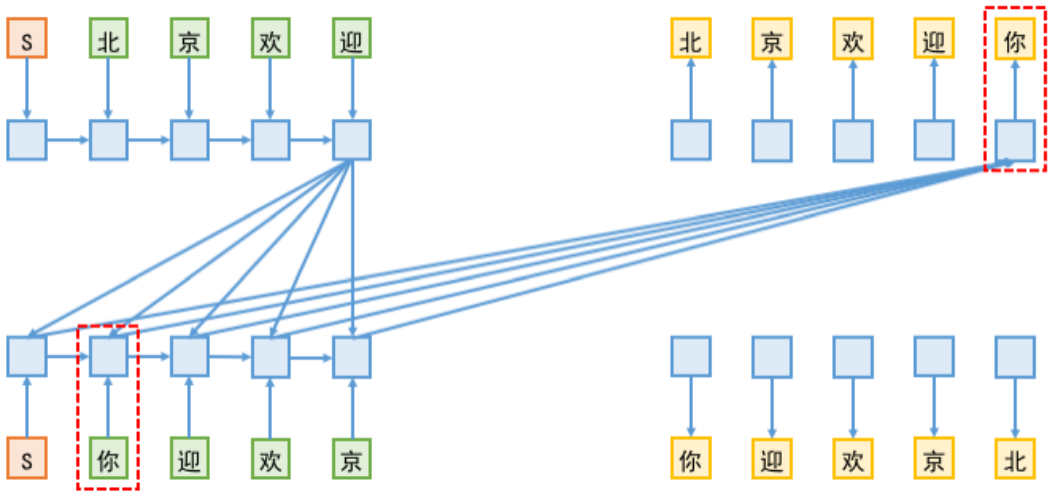

Figure 3: Information leakage diagram. During the training phase, when the L2R end is predicting a word, it actually uses the label passed into the R2L end; conversely, when the R2L end predicts a word, the same problem exists, i.e., it uses the L2R label.

An apparent phenomenon of information leakage is: In the later stages of training, the sum of the cross-entropies of the L2R and R2L directions in bidirectional decoding is even smaller than the single cross-entropy when training the unidirectional decoding model alone. This is not

because bidirectional decoding brings a huge fitting improvement, but is exactly a manifestation of information leakage.

Since information is leaked during the training process, why is such a model still useful? I think the reason is probably given in the table at the beginning of the article. Taking the same example, when the L2R end predicts the last word, it uses all the known information of the R2L end. Since the R2L end decodes word by word from right to left, according to the statistical data in the table at the beginning of the article, it is not hard to imagine that for the R2L end, the prediction accuracy of the first word (which is the last word of the sentence) should be the highest. In this way, assuming that the last word of the R2L end can indeed be predicted successfully with high accuracy, then information leakage becomes non-leakage—because information leakage occurs because we manually passed in the labels, but if the predicted result itself is consistent with the label, then leakage is no longer leakage.

Of course, the original paper also provides a strategy to mitigate this leakage problem. The general approach is to first train a version of the model using the above method, then for each training sample, use the model to generate corresponding prediction results (pseudo-labels), and then train the model again. This time, the model is trained by passing in pseudo-labels to predict the correct labels, thus maintaining the consistency of training and prediction as much as possible.

4 Summary

This article introduces and implements a bidirectional decoding mechanism for seq2seq, which symmetrizes the entire decoding process, thereby improving the generation quality to a certain extent. Personally, I think this attempt at improvement has a certain value, especially for readers who pursue formal beauty. Therefore, I have introduced it here.

In addition, the article also analyzes the possible problems of this bidirectional decoding and gives the author's own views. Readers are welcome to exchange their perspectives.