11-731 Machine Translation and Sequence to Sequence Models Assignment 1

Yinzhi Yu

March 2, 2017

1 Introduction

This project aims to train a German-English translator based on the attentional neural model. The dataset for training is IWSLT German-English dataset.

2 My Model

In this part, I give a description on how I applied the attention model into my design and some other methods I implemented. I basically follow the framework given in the pseudocode, which can be divided into following parts: training(encoding, decoding) and translation(encoding, decoding).

2.1 Training

2.1.1 Encoding

Since I used mini batch in my training process, I need to do the padding and deal with the masks. In order to make things easier, for each batch, I feed in all the training instances whose source sentences are in the same lentgh. Then I only need to do the padding for target sentences. I presorted all the training instances according to their source sentences' lengths. For each training step, I feed in batchSize sentences and for the last set of each length, the size can be smaller than batchSize. During the encoding step, I mainly used bidirectional LSTM, formed by 2 LSTMs in opposite directions: one from left to right and the other from right to left. Then for each word in the source sentence we can have 2 states for it. And the final state of a specific word is formed by concatenation of its corrsponding 2 state vectors.

2.1.2 Decoding

As for the decoding, I first feed in an endsymbol and extract an initial state. I used this state and the encoding of source sentence to compute the first context in order to further get the prediction and compute the loss for the first word. It's also important to take care of the mask. All the mask values correspoding to padding words need to be set to zero in order to get a correct loss computation.

Above is the general description of the how minibacth is used, encoding, decoding part and loss computation for training.

2.2 Translation

The encoding and decoding part is the almost the same as the training step. The only difference is that, instead of computing the loss, translation part directly get the word with the highest probability in the state if greedy search is used.

2.2.1 beam search

At first, I simply used greedy search here. Then I also tried beam search.

2.2.2 unknown word

When the unknown word get the highest probability in the translation step, things can be a little bit complicated.
First, I take a look at the attention score for each word in the source sentence. Denote the word with highest probability in source sentence by W. If W is also an unknown word in German corpos, then it's very likely that this word is a Proper Noun for some specific use. Then I just output its original German form. If not, which means that the original word which has the highest score is not an unknown word in German, then I go find the English word with the second highest probability, and output this word instead of the unknown symbol.

2.2.3 Others

I also tried adding the dropout in the LSTM, but this doesn't seem to have a great influence on my model. As for the trainer, I trained the first 10 epochs using AdamTrainer and used SimpleSGDTrainer for the rest 15 epochs. As for the hyperparameters in my final model, I used embedding vectors lengthed 256 and hidden size 512, attention size 128. The batch size I used is 32.

3 Environment

I ran my program on AWS. The instance I used is p2.xlarge. I also used the AMI provided by Graham. I used GPU and 11G memory for training. Each epoch takes about 30 minutes.

4 Result and Discussion

4.1 Result

I first trained my model without dropout. The final loss can be less than 2.4 per word on training data and less than 3.3 per word on validation data. The Bleu score I got on the test data is around 17. Then I trained with dropout set as 0.5. The loss on both the training data and validation data are a little bit higher. Before training with dropout in the whole process, I also tried to train the model with no dropout at first and then add some dropout at the later epochs. But the result turns to be very bad, worse than the other 2. That's the reason I start to train it with dropout from the start to the end in case of overfitting.

At last, I changed the embedding size from 512 to 256 in case of overfitting, and I switched the dropout to 0.2. In the translation process, I used beam size = 5. The final loss I got on the training set is 2.38 per word and the the loss on the validation set is 3.14 per word. The bleu score of the test set is 21.1313872138.

When I used the Adamtrainer, I can get a relatively fast convergence while SGDtrainer is relatively slower. But according to what Graham taught on the class, SGD can get better result in general. So during the later epochs I switched to SGD trainer.

4.2 Problems

I posted a question on piazza asking why most of my translated sentences start with "you", "he", "I", "we". And finally I found that it's because when I was computing the loss for the first word, I simply got a initial state instead of combining the initial state and the previous encoding information to get the loss. In general, it's due to falsely dealing with the start word. I also found a bug when I dealt with unknown word. I found a better replacement for unknown word and append it into the result list. But in the next round prediciting the next word, I falsely feed in the replaced word. Actually I have to feed in the original unknown symbol. Above are the 2 bugs I found after I got a bad result. After debugging, the result seems relatively better.