11-731 Machine Translation Assignment 1 Report

Ning Dong

February 26, 2017

1 Introduction

In this assignment, I created an attentional German-English translation model. The report is organized as follows. In section 2, I will introduce my implementation, particularly components and custom strategies over baseline. In section 3, experiment settings and results will be presented. Section 4 is about structure of my deliverables.

2 Implementation

2.1 Mini-batch

To fully utilize GPU power, I implement mini-batch version of the model. Key point of mini-batch in this assignment is padding and masking given we have different length of sentences. Padding and masking happen in both encoding and decoding phase.

For simplicity, I created batches where all sentences in a batch share the same length. Then I avoid padding in encoding phase(which could be tricky) with cost of slightly more batches to compute per epoch. At decoding phase, I perform padding and masking, and use the mask together with loss to update network parameters.

2.2 Unknown words replacement

At training side, I choose rare words threshold to be 5. That is, if a word appears in training set less or equal than 5 times, I mark it as <unk> (symbol of unknown words). According to my manual inspection, these words appear in test set rarely thus removing them would only increase system performance.

At output side, We have options to replace unknown words which is part of output strategies. 3 strategies I tried with are listed below.

2.2.1 Strategy 1

At certain time step when we meet <unk>, replace it with word in source language which has the largest attention score at current step.

2.2.2 Strategy 2

Maintain a list L to store words used in source sentence. At certain time step when we meet <unk>, replace it with word in source language which has the largest attention score at current step, and not in L. When size of L reaches S(which is a configurable parameter), reset L to empty.

2.2.3 Strategy 3

Import a German-English dictionary and substitute them with most similar words in target language. Actually this strategy is orthogonal to previous two and could be combined with them. However since in this assignment we are not allowed to use external data, this strategy is not implemented in final version.

Parameter	Value
Embed size	512
Hidden size	512
Attention size	128
Batch size	32
Number of layers	1
Beam Width	2
Unknown word replacement strategy	2

Table 1: Best Result Parameters

Setting ID	Embed size	Hidden size	Attention size	Batch size	Number of Layers	Final BLEU score
1	128	128	64	32	1	12.8
2	256	128	64	64	2	16.0
3	512	512	128	32	1	19.2

Table 2: Results on Different Training Parameters

2.3 Beam search

Beam search is implemented to support searching the best output sentence with custom beam widths. The model is saved after training thus we can tune parameters and try out different strategies just at output side.

3 Experiment Settings and Results

In this assignment, I use AWS p2.xlarge instance which provides powerful GPU processing cores. AMI created by instructor(ami-3fa1205f) is installed. Results below are divided into 3 parts - best result and settings, results on different training parameters including embed/hidden/attention size and network layers, comparison between different output strategies.

3.1 Best Result

I get best result on my implementation using parameters in Table 1.

After 30 epochs, I get BLEU score 19.5 on test set. It takes 2000 seconds to train one epoch. At output side, I use beam search with beam width 2 and unknown word replacement strategy 2 introduced in 2.2 above.

3.2 Results on Different Training Parameters

I trained the neural model under different parameter settings. As it costs several hours to train one model, it is impossible to do grid search exhaustively for this assignment. 3 settings and their total losses on dev set are presented in Table 2. To look at how it converges, I parsed the log and present how total loss on dev set changes in Figure 1.

For every setting I set dropout rate to be 0.5. Here greedy max method is using at output side.

3.3 Results on Different Output Strategies

By setting beam width to 1-5 and unknown word replacement strategy 1 and 2, I got result in Table 3. In my implementation the best result appears when beam width = 2.

Performance of greedy max is very close to the best. In these experiment I use model trained on setting 3 above in 3.2.

4 Deliverables

My deliverables for assignment 1 include this report, code and output files.

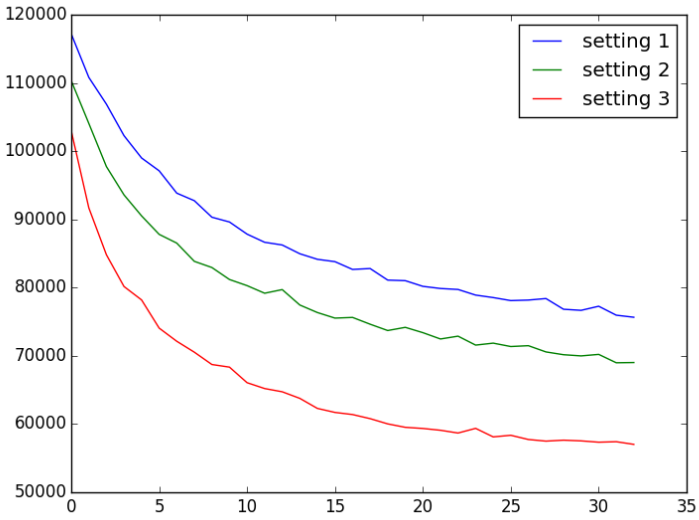

Figure 1: 'Epoch-Total loss' on dev set under 3 training parameter settings

Beam Width	BLEU-Strategy 1	BLEU-Strategy 2
1	19.2	19.3
2	19.3	19.5
3	18.9	19.3
4	18.8	18.6
5	18.8	18.9

Table 3: Results on Different Output Strategies

4.1 Code

attention_minibatch.py contains code that performs all tasks in training/saving/loading models and translating sentences.

Utility code includes computeBLEU.py to report BLEU score(using NLTK), parseLog.py to parse log files and translate.py to generate results by using trained model.

4.2 Output Files

final.log is the training log for final(best) result.

test.primary.en is my translation result on test.en-de.low.de, which reports 19.5 BLEU score.

blind.primary.en is my translation result on blind.en-de.low.de.