Natural Language Understanding with Distributed Representations - Assignment 4

Sean D Rosario - sdr 375

November 29, 2016

1 Data Preparation

The default tensorflow code downloads the WMT dataset for English-French translation. I modified `data_utils.py` so that it does not download the data and so that it reads in the data from here, which is already split into train, dev and test. No pre-processing was required as the data was already in the required format. I modified both `data_utils.py` and `translate.py` to work with Japanese and English data files instead of English and French data files

2 RNN language Model

I trained two models for Neural Machine Translation. using Tensorflow translate code. In the simple encoder-decoder model, every input has to be encoded into a fixed-size state vector, as that is the only thing passed to the decoder. To allow the decoder more direct access to the input, an attention mechanism is used which allows the decoder to peek into the input at every decoding step. Both models were trained using Stochastic Gradient Descent. ADAM optimization was attempted but did not improve results, hence is not included in this report.

The parameters I used while training are as follows:

Flag name	Value
Batch size to use during training	64
Learning rate	0.5
Number of layers in the model	3
English vocabulary size	40000
Japanese vocabulary size	40000
Batch size to use during training	64
Learning decay rate	0.99

To calculate the BLEU score, I get take every pair of examples in the test set and get the output of the source sentence and get the individual BLEU score using the nltk package between model output of the source and the actual translation. The pairwise BLEU score was calculated using `nltk.translate.bleu_score.sentence_bleu([reference], hypothesis)` , where `reference` is the actual translation and `hypothesis` is the model translation. I computed the mean BLEU score accross

The `seq2seq_model.py` has some tricks in the model such as bucketing, padding and reversed input.

2.1 Simple Encoder-Decoder Model

The simple encoder-decoder model does not use attention

I use the `embedding_rnn_seq2seq` model of Tensorflow.

The optimum model gave me these results:

Final train perplexity 5.89

- eval: bucket 0 perplexity 11.50
- eval: bucket 1 perplexity 20.53
- eval: bucket 2 perplexity 40.13
- eval: bucket 3 perplexity 102.56

Mean BLEU score: 28.63

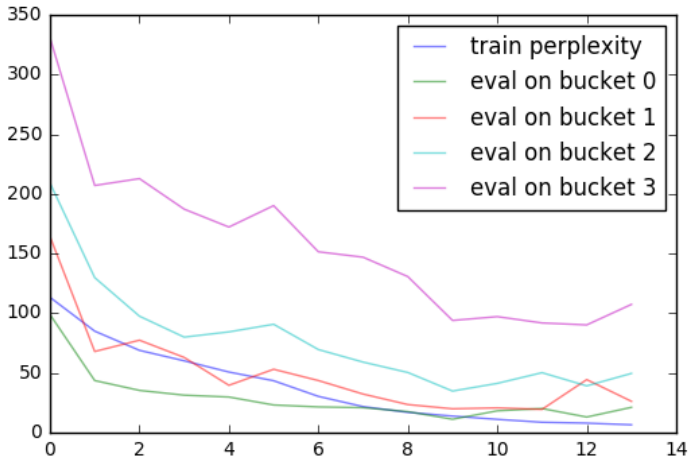

Figure 1: Perplexity vs steps for the simple encoder-decoder model without attention

2.2 Attention-based Neural Machine Translation

I use the `embedding_attention_seq2seq` model of tensorflow.

The optimum model gave me these results: Final train perplexity: 5.89

- eval: bucket 0 perplexity 11.50
- eval: bucket 1 perplexity 20.53
- eval: bucket 2 perplexity 40.13
- eval: bucket 3 perplexity 102.56

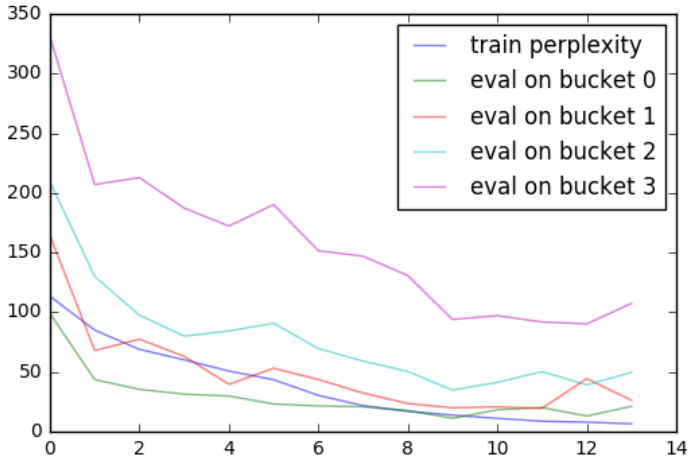

Figure 2: Perplexity vs steps for the model with attention

Mean BLEU score: 21.62

3 Summary

Adding attention intuitively improves attention. But on this dataset, the model without attention obtains a higher BLEU score than the model with attention.

There are several ways to possibly improve the performance of both models. Our data size was relatively small and adding more data to the training set would in theory improve the model

performance and reduce the possibility of the model overfitting. I could also try other optimizers apart from stochastic gradient descent and ADAM optimizer.

4 References

[1] https://www.tensorflow.org/versions/r0.11/tutorials/seq2seq/index.html