Neural Machine Translation with Attention mechanism

Arun Govind M
arungm@iisc.ac.in

1 Introduction

Machine translation aims to generate the translation of a sentence in a source language to a target language. Neural machine translation accomplishes this task by training a single neural network that can be jointly tuned to maximize the translation performance.

Attention based NMT models have been shown to greatly improve translation performance by allowing the network to selectively focus parts of the source sentence during translation.

In this assignment, neural machine translation is implemented using LSTM encoder-decoder networks. Attention mechanisms are implemented on sequence to sequence models and the consequent performance improvements are studied.

2 Data

The models are trained for English to German translation. Europarl v7, an English-German parallel corpus, from WMT14 machine translation task is used for training the models. WMT14 dev and test sets are used for validation and testing respectively.

Lines having a word count higher than 15 were removed and the resulting dataset with 508,786 sentence pairs was used for training.

3 Network architecture

3.1 Encoder-Decoder network

The encoder and decoder are both implemented as LSTM networks wherein the hidden state and cell state of the decoder LSTM are initialised as the respective final states of the encoder LSTM.

The encoder learns a fixed length context vector or "thought" vector that represents the meaning of the source sentence. The context vector is fed into the decoder network for translation.

3.2 Attention mechanism

The aforementioned fixed length context vector representation is performs poorly at 'remembering' long source sequences. Attention mechanism was devised to overcome this shortcoming. Attention can be interpreted as a vector of importance weights that allows the model to selectively focus on different parts of the source sequence while decoding as opposed to a fixed context vector.

For a given decoder state s_t, context vector c_t is the sum of encoder hidden states h_1, h_2, h_3,....,h_n weighted by the attention scores $\alpha_{t,i}$.

$$c_t = \sum_i \alpha_{t,i} h_i$$

$$\alpha_{t,i} = softmax(f_{attn}(s_t, h_i))$$

Attention mechanism differ in how alignment scores $f_{attn}(s_t, h_i)$ are defined. The following are the attention mechanisms implemented as a part of this assignment.

3.2.1 Additive attention

The first proposed attention based NMT model(Bahdanau et al., 2015)uses a one-hidden layer feed-forward network to calculate aignment scores as

$$f_{attn}(s_t, h_i) = v^T \tanh W_1 h_i + W_2 s_t$$

Where W_1 and W_2 are parameters learnt by the the attention network.

3.2.2 Multiplicative attention

(Luong et al., 2015) proposed a simplified way to calculate attention as below

$$f_{attn}(s_t, h_i) = h_i^T W_a s_t$$

where W_a is a network parameter.

3.2.3 Scaled dot attention

(Vaswani et al., 2017) proposed the following scaling to multiplicative alignment scores

$$f_{attn}(s_t, h_i) = \frac{h_i^T W_a s_t}{\sqrt{d}}$$

where W_a is a learned parameter and the parameter d in the scaling factor is the dimension of the encoder hidden states h_i

3.2.4 Key-value attention

Key value attention, maintains separate vectors for calculating alignment scores and for generating the context vector. This is done by splitting each encoder hidden state into key and value pairs. The keys k_i are used to calculate the alignments scores.

$$h_i = [k_i; v_i]$$

$$a_i = softmax(v_a^T[W_1[k_{i-L};,,,,,k_{i-1}]+W_2 k_i 1^T])$$

$$c_i = [v_{i-L}; .....; v_{i-1}]a^T$$

where 1 is a vector of ones and L is the length of the attention window.

3.3 Prediction Layer

Current decoder output s_t and context vector c_t are concatenated and fed to a two layer feed-forward neural network with softmax activation to get the probability distribution over German vocabulary. During training, one-hot encoding of the target word is used to calculate cross entropy loss associated with the distribution vector. The weights are subsequently updated in backpropagation.

During inference, the most probable word is predicted as the next word and fed back into the decoder LSTM as input. This process is repeated, until a stop token is predicted, to generate the target sequence.

4 Implementation

Attention based NMT models are implemented in PyTorch.

4.1 Data Preprocessing

From the English-German parallel corpus, sentences with word count greater than 15 are removed. The remaining 508,786 are tokenized and used to build vocabulary. Keras' Text tokenization utility class is used to do the same.

'¡start¿' and '¡end¿' token are inserted at either end of every sentence and OOV words are replaced with '¡UNK¿' tokens. The sentences are batched and padded.

model	BLEU held-out	BLEU WMT
seq2seq	4.51	1.18
seq2seq + additive	12.8	8.52
seq2seq + multiplicative	8.36	3.83
seq2seq + scaled dot	8.15	3.79

4.2 Training

Batches of padded input and target sequences are fed to the encoder and decoder LSTMs. The output distribution from the prediction layer is evaluated against one-hot encoding of target word and cross entropy loss is calculated. Mean loss is minimized by Adam optimizer.

4.3 Inference

The output of the prediction layer is used to predict the next word of the target sequence. The word with the highest probability mass is the model's predicition for the next word. The same is fed back into the decoder as input for the next time step. This process is repeated until the stop token in predicted. Some models tend to predict the same sequence of words indefinitely without terminating and therefore the max prediction length is capped at 30.

4.4 Metric

BLEU score is used to evaluate the performance of the models.

5 Experiments

A vanilla seq2seq model is trained to establish baseline for comparisons. Each attention variant is then trained for hidden state dimensions 128 and 64 respectively. This is done to study the effect of hidden state dimension on effectiveness. While additive and multiplicative attentions are expected to perform comparably for low dimensions, for larger hidden state dimensions, additive attention is expected to outperform multiplicative attention. The models are tested on WMT14 dev set and a held-out data from Europarl v7 dataset.

6 Observations and Conclusions

As seen from the results, adding attention results in substantial gains in performance of the model for Machine Translation task. Additive attention variant trumps multiplicative and scaled-dot. This could be due to high dimensionality of the hidden

states, viz. 128. However, decreasing the dimensionality of decoder hidden states to 64 did not close the gap between additive and scaled dot attention variants.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

scores.png

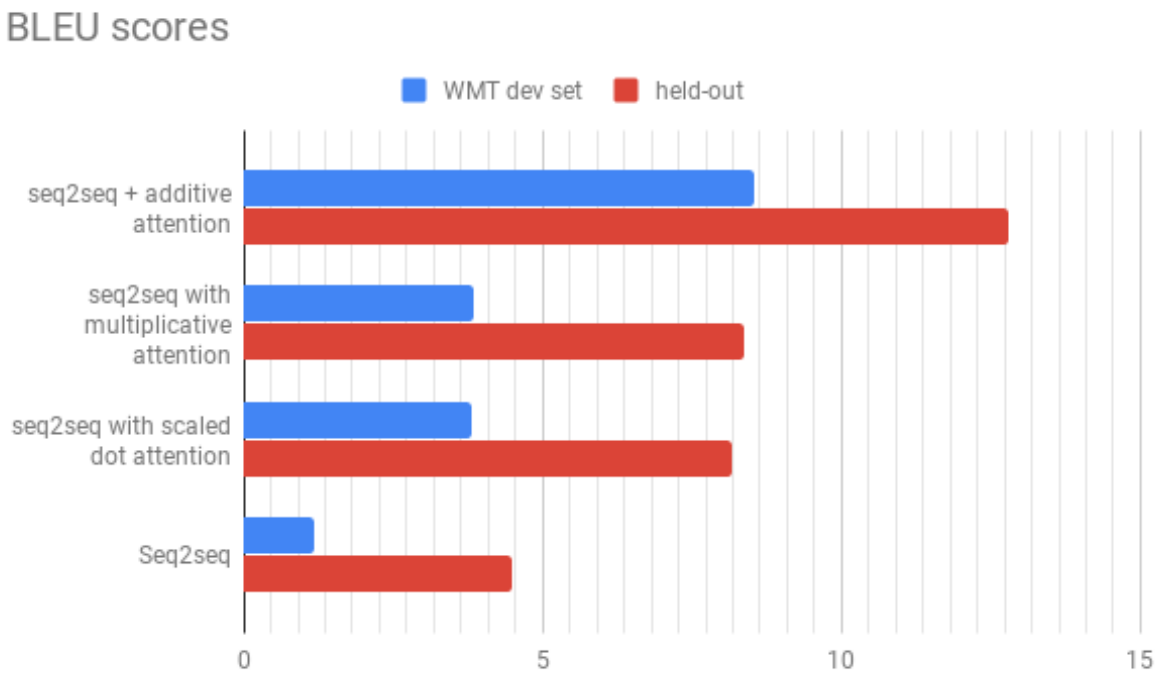

Figure 1: BLEU scores

score vs hidden size.png

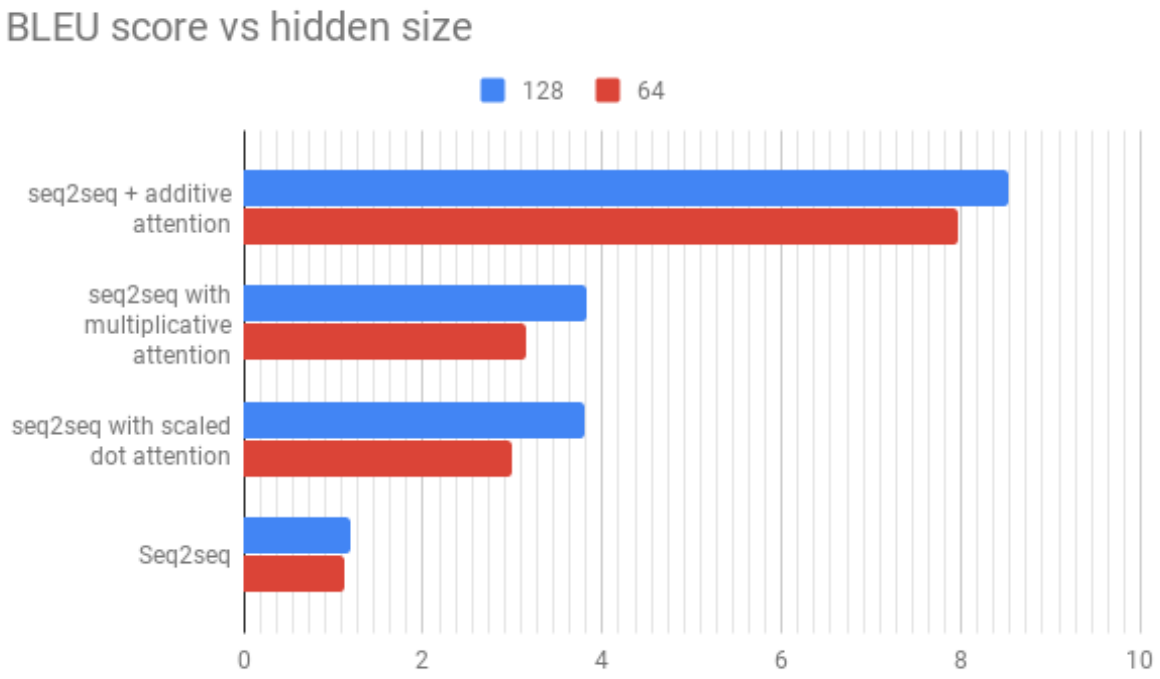

Figure 2: Effect of hidden size on BLEU scores