Effective Approaches to Attention-based Neural Machine Translation

1 Citation

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025 (2015).

`https://arxiv.org/pdf/1508.04025.pdf`

2 Abstract

We create a global and local attention system for Neural Machine Translation. The local system is 5 BLEU points better than the global one. Our ensemble system sets state of the art on English to German translation with 25.9 BLEU points.

3 Introduction

In Neural Machine Translation (NMT), we use a stack of LSTMs to encode a sentence into a sequence of vectors. Then, a stack of LSTMs decodes an attention vector (one per output word, computed by combining the encoded vectors) to produce the output words. The original NMT model uses a kind of global attention. We develop another global attention model and a local attention model that is more computationally efficient and accurate.

4 Neural Machine Translation

We aim to predict output sequence $y = y_1, ..., y_m$ from input sequence $x = x_1, ..., x_n$. The encoder produces a representation $\mathbf{s}$.

$$\log p(y|x) = \sum_{j=1}^{m} \log p(y_j|y_{<j}, s)$$

where, for some transformation g:

$$p(y_j|y_{<j}, \mathbf{s}) = \text{softmax}(g(\mathbf{h}_j))$$
$$\mathbf{h}_j = f(\mathbf{h}_{j-1}, \mathbf{s})$$

The training objective is (for dataset $\mathbb{D}$):

$$J_t = \sum_{(x,y)\in\mathbb{D}} -\log p(y|x)$$

5 Attention Based Models

Given a hidden state of the decoder $\mathbf{h}_t$, we combine it with a context vector $\mathbf{c}_t$ to get an attentional hidden state $\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c[\mathbf{c}_t, \mathbf{h}_t])$. Our output is then $p(y_t|y_{<t}, x) = \text{softmax}(\mathbf{W}_s\tilde{h}_t)$. We can compute $\mathbf{c}_t$ with local or global attention.

First, let's consider global attention, which works like this (let $\mathbf{h}_t$ represent a decoder hidden state and $\bar{\mathbf{h}}_t$ represent an encoder hidden state).

$$\mathbf{a}_t(s) = \text{align}(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \frac{\exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s))}{\sum_{s'} \exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}'_s))}$$

where we consider three possible scores: dot ($\mathbf{h}_t^T\bar{\mathbf{h}}_s$), general ($\mathbf{h}_t^T\mathbf{W}_a\bar{\mathbf{h}}_s$) and concat ($\mathbf{v}_a^T \tanh(\mathbf{W}_a[\mathbf{h}_t; \bar{\mathbf{h}}_s])$)

We then compute $\mathbf{c}_t$ as a weighted average (over s) of the alignment vectors, $\mathbf{a}_t(s)$, with learned weighting. Note, we also experimented with an alignment function that is totally independent of the source: $\mathbf{a}_t = \text{softmax}(\mathbf{W}_a\mathbf{h}_t)$. Unlike the regular NMT system, we use the decoder hidden state, have a simpler computation, and try different score functions.

The problem with global attention is that we must average over each hidden vector produced by the encoder (one per input word in the sequence). If the input sequence is long, this can be compute intensive. To avoid this, we use local attention where, for decoder hidden vector $\mathbf{h}_t$, we only consider encoder hidden vectors between $[p_t - D, p_t + D]$. How do we pick p_t? One approach is to simply do $p_t = t$. A better approach is to do (for parameters $\mathbf{W}_p, \mathbf{v}_p$ and input length S):

$$p_t = S \cdot \text{sigmoid}(\mathbf{v}_p^T \tanh(\mathbf{W}_p\mathbf{h}_t))$$

Additionally, we can encode a preference for encoder hidden vectors near p_t with a Gaussian centered at p_t (we set $\sigma = D/2$):

$$\mathbf{a}_t(s) = \text{align}(\mathbf{h}_t, \bar{\mathbf{h}}_s) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right)$$

We'd also like to consider the previous attention decision when making the current attention decision. To do this, we augment the decoder input with the $\tilde{\mathbf{h}}_t$ from the previous time step. This is called input feeding.

6 Experiments

We evaluate on the WMT English to German task and evaluate with BLEU score. Our dataset has 4.5M sentences (about 110M words in each langugage). Our vocabularies are the 50K most common words in each language (everything else is denoted with a special "unknown" token). We use 4-layer, 1000-cell LSTMs with 1000-dimensional embeddings. Parameters are uniformly sampled from $[-0.1, 0.1]$, SGD (start with learning rate 1) is our optimization algorithm, and learning rate is halved every epoch after 5 epochs. We use 128-sentence minibatches where each minibatch has sentences of similar length to avoid making short sentences wait for long ones to finish processing. We clip gradients when norm exceeds 5. We set $D = 10$. We also experimented with dropout (probability 0.2), which required 12 epochs for training and halving learning rate after epoch 8. It takes 7-10 days to train our model on an NVIDIA Tesla at 1K target words a second.

We evaluate against the state of the art WMT English-German model (a phrase based model) and RNNSearch (a previous NMT). We get BLEU boosts from reversing input sentence (+1.3), dropout (+1.4), global attention (+2.8), input feeding (+1.3), replacing global attention with local attention where we predict p_t (+0.9), unknown replacement technique (+1.9). Our final ensemble gets 25.9 BLEU, which is state of the art. We also tried German-English translation and did really well there.

7 Analysis

Input feeding, local attention, and dropout all decrease the final error in the learning curves. We beat all previous models on every sentence length. The "dot" score is better for global attention and the "general" score is better for local attention. The best model is local attention where we predict p_t. Given an alignment dataset, we computed the Alignment Error Rate and find that our local attention with predicted p_t scores best. Sample phrases look good.

8 Conclusion

We present richer global and local attention mechanisms for NMT and set state of the art on English-German translation.