HW4: All about Attention

Jiafeng Chen Yufeng Ling Francisco Rivera

April 3, 2019

1 Introduction

In this writeup we consider natural language inference–given a premise and a hypothesis, can we determine the entailment and contradiction relationship between them. The key to the model is the attention architecture, which serves to decompose the problem into aligned subphrases. Not only does this design make training parellelizable, it also significantly reduces the number of parameters while delivering state-of-the-art results.

2 Problem Description

In this writeup, we consider the problem of natural language inference. Let

$$\boldsymbol{a} = (a_1, \ldots, a_{\ell_a}) \tag{1}$$

be the premise of length ℓ_a and let

$$\boldsymbol{b} = (b_1, \ldots, b_{\ell_b}) \tag{2}$$

be the hypothesis of length ℓ_b. Each $a_i, b_j \in \mathbb{R}^d$ is a word embedding vector of dimension d. Our goal is to, given the input pair $\boldsymbol{a}, \boldsymbol{b}$, predict the relationship $y \in \{y_1, \ldots, y_C\}$ where C is the number of output classes.

3 Model and Algorithms

3.1 Decomposable Attention Model

This section follows the architecture of Parikh et al. (2016).

3.1.1 Vanilla model

We first look at the most basic approach that is the foundation of this architecture. We start by setting the inputs $\bar{\mathbf{a}}, \bar{\mathbf{b}}$ to be the premise and hypothesis $\mathbf{a}, \mathbf{b}$ themselves. We generate attention weight matrix by softmaxing over

$$e_{ij} := F'(\bar{a}_i, \bar{b}_j) = F(\bar{a}_i)^\intercal F(\bar{b}_j). \tag{3}$$

Note that we made the simplification of setting F' to be the dot product of $\bar{a}_i$ and $\bar{b}_j$ through the same feed-forward neural network, which reduces the number of operations from $O(\ell_a \times \ell_b)$ to $O(\ell_a + \ell_b)$. The attended phrases are then

$$\begin{aligned} \beta_i &:= \sum_{j=1}^{\ell_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_b} \exp(e_{ik})} \bar{b}_j, \\ \alpha_j &:= \sum_{i=1}^{\ell_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_a} \exp(e_{kj})} \bar{a}_i. \end{aligned} \tag{4}$$

A key implementation detail is that we need to apply masking to the $\{e_{ij}\}$ matrix before softmaxing to avoid putting attention on padding.

Next, we compare the aligned attended phrases to the original ones by concatenating them and applying a feed-forward neural network G.

$$\begin{aligned} \mathbf{v}_{1,i} &:= G([\bar{a}_i, \beta_i]) \quad \forall i \in [1, \ldots, \ell_a], \\ \mathbf{v}_{2,j} &:= G([\bar{b}_j, \alpha_j]) \quad \forall j \in [1, \ldots, \ell_b]. \end{aligned} \tag{5}$$

We then apply sum-over-time pooling, with padding masked out, to generate the penultimate vectors

$$\mathbf{v}_1 = \sum_{i=1}^{\ell_a} \mathbf{v}_{1,i}, \qquad \mathbf{v}_2 = \sum_{j=1}^{\ell_b} \mathbf{v}_{2,j}. \tag{6}$$

Finally, we apply a feed-forward neural network H to the concatenated vectors to generate the unnormalized predictions for each class

$$\widehat{y} = H([\mathbf{v}_1, \mathbf{v}_2]) \in \mathbb{R}^C. \tag{7}$$

The prediction is $\widehat{y} = \arg\max_i \widehat{\mathbf{y}}_i$. In the training of this model, we use the multi-class cross-entrophy loss as the loss function.

$$L(\theta_F, \theta_G, \theta_H) = \frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} y_c^{(n)} \log \frac{\exp(\widehat{y}_c)}{\sum_{c'=1}^{C} \exp(\widehat{y}_{c'})}. \tag{8}$$

3.1.2 Intra-Sentence Attention

We can improve the model by incorporating intra-sentence attention. Instead of having $(\overline{\mathbf{a}}, \overline{\mathbf{b}}) = (\mathbf{a}, \mathbf{b})$, we add self-attention to the input. We let the unnormalized attention weights be

$$f_{ij} = F_{\text{intra}}(a_i)^\intercal F_{\text{intra}}(a_j), \tag{9}$$

where F_{intra} is a feed-forward network. We then create the self-aligned phrases

$$a'_i = \sum_{j=1}^{\ell_a} \frac{\exp(f_{ij} + d_{i-j})}{\sum_{k=1}^{\ell_a} \exp(f_{ik} + d_{i-k})} a_j. \tag{10}$$

In the above equation, $d_{i-j} \in \mathbb{R}$ is the bias term based on distance, which is shared throughout sentences. Moreover, we bucket the terms such that all distances greater than 10 have the same bias. In the end, we use $\overline{a}_i = [a_i, a'_i]$ and $\overline{b}_j = [b_j, b'_j]$ as inputs.

3.2 Latent Variable Mixture Model

4 Experiments

model name	specifications
`Seq2Seq`	100 embedding, 100 hidden, no dropout
`Seq2SeqAttn`	Bi-directional LSTM, 100 embedding, 100 hidden, no dropout

Table 1: Both models used only one layer in the decoder and trained with Adam and learning rate 10^{-3} *over 10 epochs. For* `Seq2Seq`*, decrease in validation loss flattened at epoch 8. For* `Seq2SeqAttn`*, training stopped at epoch 7 after loss starting going up.*

model name	Loss	Perplexity	BLEU
`Seq2Seq`	3.23	25.36	6.32
`Seq2SeqAttn`	2.71	15.05	17.77
`Megatron`	2.7	15	9.68
`Soundwave`	2.8	16	7.08

Table 2: Performance metrics for different models

References

Parikh, A. P., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

A Model implementation