Attention as a Latent Variable Model

February 20, 2018

1. Language as a Latent Variable

1. Language as a Latent Variable

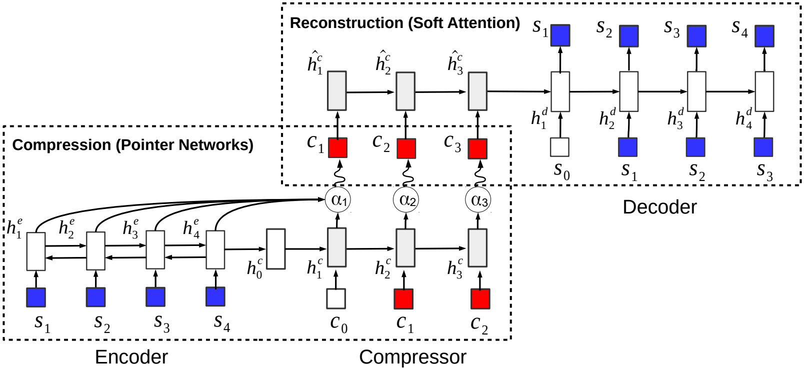
Overview of Paper

- Semi-supervised summarization
- Builds on Kingma et al. [2014]'s M2 (the semisupervised version)
- But uses a sequence of words as the latent representation

The Idea for Extractive Summarization

- Start with source sentence $\mathbf{x} =$ 'I wish I could love dogs but I just hate them'
- Sample a summary $\mathbf{y} =$ 'I love dogs' by picking words from the source
- Reconstruct the source sentence given the summary using an attentive decoder
- Use the probability of the source sentence under the reconstruction decoder (and a couple other terms) as signal for how good the summary was

Auto-Encoding Sentence Compression



Note: Some of the parameters of the inference network are used as the decoder network's encoder, but the shared parameters are not updated using gradients from the decoder

Auto-Encoding Sentence Compression

- The inference network $q_\lambda(\mathbf{y} \mid \mathbf{x})$ uses hard attention at every timestep to select a source token
- Use a bidirectional source encoder on $\mathbf{x}$ to get source embedding matrix $H^e = \mathrm{BRNN}(\mathbf{x})$, whose ith element is the vector $\mathbf{h}^e_i$.
- Select source token $\mathbf{x}_i$ with a pointer network 'compressor'

$$\mathbf{h}^c_j = \mathrm{RNN}(\mathbf{h}^c_{j-1}, \mathbf{y}_{j-1}) \tag{1}$$

$$\mathbf{u}_j = \mathrm{attention}(\mathbf{h}^c_j, H^e) \tag{2}$$

$$\mathbf{y}_j \sim \mathrm{Cat}(\mathbf{u}_j) \tag{3}$$

ASC Continued

- Let H^c be the concatenation of all the compressor hidden states
- The decoder $p_\theta(\mathbf{x} \mid \mathbf{y})$ is a conditional language model that attends over the hidden states H^c to reconstruct the original source sentence $\mathbf{x}$

$$\mathbf{h}_k^d = \mathrm{RNN}(\mathbf{h}_{k-1}^d, \mathbf{x}_{k-1}) \tag{4}$$

$$\mathbf{v}_k = \mathrm{attention}(\mathbf{h}_k^d, H^c) \tag{5}$$

$$\mathbf{d}_k = \mathbf{v}_k^T H^c \tag{6}$$

$$p_\theta(\mathbf{x}_k \mid \mathbf{x}_{<k}, \mathbf{y}) = \mathrm{softmax}(W\mathbf{d}_k) \tag{7}$$

- (7) is quite strange, since it does not use the recurrent state. My guess is that it is a mistake from following the pointer network paper too closely (this is equation (10) in the paper).

Details and Recap

- The attention is the attention formulation from Vinyals et al. [2015], which is pretty close to the 'general' attention in Luong et al. [2015]

$$\text{attention}(q, C) = \text{softmax}(\mathbf{v}^T \tanh(Wq + VC)) \tag{8}$$

- $p_\theta(\mathbf{x} \mid \mathbf{y})$ is the reconstructive attention based decoder
- $q_\lambda(\mathbf{y} \mid \mathbf{x})$ is the summarizing pointer network, and we omit the conditioning on $\mathbf{x}$ when convenient (randomly)
- $p(\mathbf{y})$ is a language model prior

Marginal Likelihood

- The inference network's parameters will be denoted by λ and the decoder network's by θ
- As usual, the marginal likelihood is intractable

$$\log p(\mathbf{x}) = \log \sum_{\mathbf{y}} p_\theta(\mathbf{x}, \mathbf{y}) \tag{9}$$

since we cannot enumerate all possible summaries, even if they are extractive

Objective

- So we lower bound it with Jensen's inequality and maximize the ELBO $\mathcal{L}$

$$
\begin{aligned}
\log \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}) &= \log \sum_{\mathbf{y}} q_\lambda(\mathbf{y}) \frac{p_\theta(\mathbf{x}, \mathbf{y})}{q_\lambda(\mathbf{y})} && (10) \\
&= \log \mathbb{E}_{\mathbf{y} \sim q_\lambda(\mathbf{y})} \left[\frac{p_\theta(\mathbf{x}, \mathbf{y})}{q_\lambda(\mathbf{y})} \right] && (11) \\
&\geq \mathbb{E}_{\mathbf{y} \sim q_\lambda(\mathbf{y})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{y})}{q_\lambda(\mathbf{y})} \right] && (12) \\
&= \mathbb{E}_{\mathbf{y} \sim q_\lambda(\mathbf{y})} \left[\log p_\theta(\mathbf{x} \mid \mathbf{y}) \right] && (13) \\
&\quad - D_{KL}(q_\lambda(\mathbf{y}) \| p(\mathbf{y})) \\
&= \mathcal{L}
\end{aligned}
$$

Training Details

- The gradient of the ELBO with respect to the reconstructive decoder only depends on $p_\theta(\mathbf{x} \mid \mathbf{y})$

$$\mathcal{L} = \underbrace{\mathbb{E}_{\mathbf{y} \sim q_\lambda(\mathbf{y})} \left[\log p_\theta(\mathbf{x} \mid \mathbf{y})\right]}_{\text{1. Reconstruction}} - \underbrace{KL[q_\lambda(\mathbf{z}) \| p(\mathbf{z})]}_{\text{2. Regularization towards prior}}$$

- It's given by term 1

$$\nabla_\theta \mathcal{L} = \nabla_\theta \mathbb{E}_{\mathbf{y} \sim q_\lambda(\mathbf{y})} \left[\log p_\theta(\mathbf{x} \mid \mathbf{y})\right] \tag{14}$$

$$\approx \frac{1}{M} \sum_m \nabla_\theta \log p_\theta(\mathbf{x} \mid \mathbf{y}^{(m)}) \tag{15}$$

where M sample summaries are generated through ancestral sampling

Training Details

- The gradient with respect to the inference network requires REINFORCE
- We rewrite the ELBO

$$\mathcal{L} = \mathbb{E}_{\mathbf{y} \sim q_\lambda(\mathbf{y})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{y})}{q_\lambda(\mathbf{y})} \right] \tag{16}$$

$$= \mathbb{E}_{\mathbf{y} \sim q_\lambda(\mathbf{y})} \left[\log p_\theta(\mathbf{x} \mid \mathbf{y}) + \log p(\mathbf{y}) - \log q_\lambda(\mathbf{y}) \right] \tag{17}$$

$$= \mathbb{E}_{\mathbf{y} \sim q_\lambda(\mathbf{y})} \left[l(\mathbf{x}, \mathbf{y}) \right] \tag{18}$$

- So we have $l(\mathbf{x}, \mathbf{y}) = \log p_\theta(\mathbf{x} \mid \mathbf{y}) + \log p(\mathbf{y}) - \log q_\lambda(\mathbf{y})$

REINFORCE

- Recall the score function gradient estimator

$$p(\mathbf{x})\nabla \log p(\mathbf{x}) = \nabla p(\mathbf{x}) \tag{19}$$

- We use this to find an approximation of

$$\begin{aligned}
\nabla_\lambda \mathcal{L} &= \nabla_\lambda \mathbb{E}_{\mathbf{y}\sim q_\lambda(\mathbf{y})}\left[l(\mathbf{x},\mathbf{y})\right] && (20)\\
&= \sum_{\mathbf{y}} l(\mathbf{x},\mathbf{y})\nabla_\lambda q_\lambda(\mathbf{y}) && (21)\\
&= \sum_{\mathbf{y}} q_\lambda(\mathbf{y}) l(\mathbf{x},\mathbf{y})\nabla_\lambda \log q_\lambda(\mathbf{y})) && (22)\\
&= \mathbb{E}_{\mathbf{y}\sim q_\lambda(\mathbf{y})}\left[l(\mathbf{x},\mathbf{y})\nabla_\lambda \log q_\lambda(\mathbf{y})\right] && (23)
\end{aligned}$$

Details

- They also train a baseline to predict $l(\mathbf{x}, \mathbf{y})$ for variance reduction
- They use a variant of KL annealing, and augment the loss as follows

$$l(\mathbf{x}, \mathbf{y}) = \log p_\theta(\mathbf{x} \mid \mathbf{y}) + \lambda(\log p(\mathbf{y}) - \log q_\lambda(\mathbf{y}))$$

  with $\lambda = 0.1$ without justification, but note that increasing λ results in shorter summaries $\mathbf{y}$ since the prior $p(\mathbf{y})$ prefers shorter sequences

Questions

- Why don't we use

$$\log \sum_{\mathbf{y}} p(\mathbf{y}) p_\theta(\mathbf{x} \mid \mathbf{y}) \geq \sum_{\mathbf{y}} \mathbb{E}_{\mathbf{y} \sim p(\mathbf{y})} \left[\log p_\theta(\mathbf{x} \mid \mathbf{y}) \right]$$

  instead of introducing $q_\lambda(\mathbf{y})$ if we're using REINFORCE anyway?

- Which parts of the reward

$$l(\mathbf{x}, \mathbf{y}) = \log p_\theta(\mathbf{x} \mid \mathbf{y}) + \log p(\mathbf{y}) - \log q_\lambda(\mathbf{y} \mid \mathbf{x})$$

  can we decompose to try to lower variance a bit more?

Connection to MIXER

- lol

Title

- lol

D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-supervised learning with deep generative models. *CoRR*, abs/1406.5298, 2014. URL http://arxiv.org/abs/1406.5298.

M.-T. Luong, H. Pham, and C. D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of EMNLP*, 2015.

O. Vinyals, M. Fortunato, and N. Jaitly. Pointer Networks. In *Proceedings of NIPS*, 2015.