Latent Constituents via Self-Attention

August 8, 2018

1 Introduction

2 Problem

We would like to learn a generative model over source sentences $\mathbf{x} = \{x_0, x_1, \ldots\}$, using a distribution over latent trees $\mathbf{z}$. We are primarily interested in the posterior distribution over trees given a sentence, $p(\mathbf{z} \mid \mathbf{x})$. The performance of the generative model is secondary since we would like to analyze the output of the unsupervised model for any linguistic regularities, and use the model as a testbed for linguistic hypotheses (of which I have none at the moment).

(Shen et al., 2018) Yin et al. (2018)

3 PRPN (Shen et al., 2018)

The Parsing-Reading-Predict Network (Shen et al., 2018) is an approach to unsupervised tree induction that uses a soft approximation to a latent variable to a degree of success. The paper is inspired by the following hypothesis: given a binary tree, a token at index t only requires information up to index l_t that satisfies either of the following conditions:

(a) the token at l_t is the leftmost sibling of the token at t

(b) or, if the token at t is a leftmost child, l_t points to its parent's left sibling's leftmost child.

Although the hypothesis itself is not tested in the implementation and serves only as inspiration, the model does see empirical success in the task of language modeling. The model is realized through the following insight: given a ranking of tokens $\mathbf{x} = \{x_0, \ldots, x_T\}$, recursively splitting $\mathbf{x}$ using the following procedure induces a binary tree where the first token to the left of token x_t that has higher rank, denoted x_{l_t}, also satisfies condition (a) or (b): given token i with the next highest rank, create a subtree $(x_{<i}, (x_i, x_{>i}))$ and recursively perform this procedure until only terminal nodes remain.

[Example on board]

3.1 The Model

Let x_t be the current token, $\mathbf{x}_{0:t-1}$ all previous tokens, z_t the prediction for the current score, and $\tilde{\mathbf{z}}_{0:t-1} \in \mathbb{R}_+^{t-1}$ be all previous scores. The model takes the form a language model, where the distribution over the next token

$$p(x_t \mid \mathbf{x}_{0:t-1}, \tilde{\mathbf{z}}_{0:t-1}, z_t) = f([h_{l_t:t-1}, h_t])$$

is parameterized by either a linear projection or additional residual blocks applied to a concatenation of the output of modified LSTMN (LSTM Memory-Network) h_t and a convex combination of the previous LSTMN outputs (i.e. attention over $h_{l_t:t-1}$). f is referred to as the Predict Network.

3.1.1 LSTMN

The LSTMN, referred to as the Reading Network, is as follows: at each time step, the hidden and cell input are given by

$$\{\tilde{h}_t, \tilde{c}_t\} = \sum_{i=1}^{t-1} s_i^t \{h_i, c_i\},$$

a convex combination of the previous hidden and cell outputs. Then

$$h_t, c_t = \texttt{LSTM}(\tilde{h}_t, \tilde{c}_t).$$

The attention is computed in the same way as detailed in section 3.1.2. The input to the LSTMN module is either the previous token or the output of a previous LSTMN layer.

3.1.2 Attention

As all attention modules in the PRPN network use the same formulation of attention we will use y to refer to the input of the attention module, $\boldsymbol{g}$ the gate coefficients, $\boldsymbol{m}$ as the memories to attend over, and $\boldsymbol{s}$ as the attention coefficients. First the key is computed using a linear projection of the input $k = W_k y$. Then the intermediate scores are computed

$$\tilde{s}_i = \text{softmax}\left(\frac{m_i k^T}{\sqrt{\delta_k}}\right),$$

where δ_k is the dimension of the hidden state. Finally, the gate coefficients are normalized and used to prevent the attention from extending too far in the past.

$$s_i = \frac{g_i}{\sum_j g_j} \tilde{s}_i,$$

where the gate coefficients $\boldsymbol{g}$ are detailed in section 3.1.3.

3.1.3 Gating Self-attention

Recall that the gates g_i^t used modulate self-attention also indirectly induce tree structure (due to the previously stated insight that modulating self-attention using a ranking induces a binary tree). At every timestep for token x_t, Shen et al. (2018) define a latent variable l_t which corresponds to the index of the token satisfying either condition (a) or (b). We then would have

$$g_i^t = \begin{cases} 1, & l_t \leq i < t \\ 0, & \text{otherwise.} \end{cases}$$

where we allow the model at timestep t to only attend up to the token at l_t. We would then renormalize the attention accordingly. In order to calculate the gates g_i^t we must model $p(l_t \mid \mathbf{x}_{0:t})$ (Shen et al. (2018) use the posterior distribution in their notation). They propose to model

$$p(l_t = i \mid \mathbf{x}_{0:t-1}) = (1 - \alpha_i^t) \prod_{j=i+1}^{t-1} \alpha_j^t$$

using a stick-breaking process. Rather than resorting to approximate inference, they instead use the expected value of g_i^t

$$\mathbb{E}\left[g_i^t\right] = F_{l_t}(l_t \leq i \mid \mathbf{x}_{0:t-1}) = \prod_{j=i+1}^{t-1} \alpha_j^t.$$

The $\alpha_j^t = \sigma(d_t - d_j)$, where σ is the hard sigmoid function. This is not stated in the paper (there are a couple off-by-one) errors, but it must be that $g_{t-1}^t = 1$. The $d_j \in \mathbb{R}_+$ are given by a CNN over the token embeddings.

The network used to compute the gates is called the Parsing Network. When computing $p(x_t \mid \ldots)$, we cannot use the posterior score d_t, so an estimate is used based on the previous k words, where k is the kernel width of the convolution. However, afterwards Shen et al. (2018) use the posterior score for all d_i when $i < t$.

4 Comparison to CCM (Klein and Manning, 2002a)

(Klein and Manning, 2002b) (Golland et al., 2012; Huang et al., 2012)

5 Comparison to StructVae (Yin et al., 2018)

6 Alternative Latent Ranking Model

6.1 Generative Model

PRPN (Shen et al., 2018) choose to view l_t as a latent variable. There are at least a few other choices:

- Model the scores $d_t \sim \mathcal{N}(\mu_t, \sigma_t)$ or $d_t \sim$ Gamma and use the Plackett-Luce ranking distribution.
- Model the comparisons $p(d_t < d_i)$ as order statistics of Gammas.
- Model the permutation matrix $Z \sim \mathcal{B}_n$.
- Model $l_t \sim$ Cat.

Parameterize with $d_t \sim \mathcal{N}(\mu_t, \sigma_t)$. Reparameterize comparisons with gumbel softmax? Leave all self attentions as is?

- $p(z)$
- $p(x|z)$

7 Training and Inference

References

Dave Golland, John DeNero, and Jakob Uszkoreit. A feature-rich constituent context model for grammar induction. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 2: Short Papers*, pages 17–22. The Association for Computer Linguistics, 2012. ISBN 978-1-937284-25-1. URL `http://www.aclweb.org/anthology/P12-2004`.

Yun Huang, Min Zhang, and Chew Lim Tan. Improved constituent context model with features. In *PACLIC*, pages 564–573. PACLIC 26 Organizing Committee and PACLIC Steering Committee / ACL / Faculty of Computer Science, Universitas Indonesia, 2012.

Dan Klein and Christopher D. Manning. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 128–135. ACL, 2002a. URL `http://www.aclweb.org/anthology/P02-1017.pdf`.

Dan Klein and Christopher D. Manning. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 128–135, Stroudsburg, PA, USA, 2002b. Association for Computational Linguistics. doi: 10.3115/1073083.1073106. URL `https://doi.org/10.3115/1073083.1073106`.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rkgOLb-0W`.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. StructVAE: Tree-structured latent variable models for semi-supervised semantic parsing. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia, July 2018. URL `https://arxiv.org/abs/1806.07832v1`.