Latent Constituents via Self-Attention

September 7, 2018

1 PRPN (Shen et al., 2018)

The Parsing-Reading-Predict Network (Shen et al., 2018) is an approach to unsupervised constituency tree induction that uses a soft approximation to a latent variable to a degree of success. PRPN achieves this by learning a scoring model that limits the range of self-attention in a self-attentive language model.

2 The Language Model

Let $\mathbf{x}$ be all tokens and $\mathbf{l}$ all latent variables with each $l_t \in \{0, \dots, t-1\}$ a categorical RV over all previous indices. In addition to modeling the tokens, Shen et al. (2018) introduce a latent variable at every timestep l_t. They decompose the joint distribution

$$p(\mathbf{x}, \mathbf{l}) = \prod_t \underbrace{p(x_{t+1} \mid l_t, \mathbf{x}_{\leq t}, \mathbf{l}_{<t})}_{\text{Self-attentive LM}} \underbrace{p(l_t \mid \mathbf{x}_{\leq t})}_{\text{Attention limit}} . \tag{1}$$

2.1 $p(x_{t+1} \mid l_t, \mathbf{x}_{\leq t}, \mathbf{l}_{<t})$

The first term takes the form of a self-attentive language model, where the distribution over the next token is

$$p(x_{t+1} \mid l_t, \mathbf{x}_{\leq t}, \mathbf{l}_{<t}) = \text{softmax}(f(\boldsymbol{m}_{<t}, l_t))$$

where f is either a linear projection or additional residual blocks, composed with the output of an LSTMN (LSTM Memory-Network), as well as the latent variable l_t. The LSTMN attends over the previous $\boldsymbol{m}_{<t}$ to produce m_t, the current hidden state. This is referred to as the Predict Network.

The attention mechanism is defined similarly throughout their network. The recurrent input to the current timestep is a convex combination of all previous outputs of the current layer. Let $\boldsymbol{m}_{<t}$ be the concatenated output of all previous timesteps, then the recurrent input would be

$$c = \sum_{i=0}^{t-1} s_i(l_t) m_i \tag{2}$$

and the output of the layer at the current timestep would be $m_t = LSTM(c, x_t)$ where s_i is the attention coefficient for that index at timestep t and x_t is the input.

The attention coefficients are a renormalized weighted sum of the dot-product attention we are used to. Let $\tilde{\boldsymbol{s}}_t \in \Delta^{t-2}$, the $t-2$ simplex, be the output of a softmax over the dot-product of memories $\boldsymbol{m}_{<t}$ and query k_t which is a linear function of the input x_t. We then define

$$s_i(l_t) = \frac{g_i(l_t)\tilde{s}_i}{\sum_j g_j(l_t)\tilde{s}_j}, \tag{3}$$

where $g_i(l_t) = \mathbf{1}(l_t \leq i)$. We have a set of these **per timestep**. Note that although we define $p(l_t)$ in the following section, we will end up using $\mathbb{E}_{l_t}[s_i(l_t)]$ as a soft approximation to the latent variable model. The LSTMN is referred to as the Reading Network.

[Work through Figure 1]

	0	1	2	3	4
raw attn $\tilde{s}$	0.4	0.1	0.3	0.2	
gates $g \mid l_4 = 2$	0	1	1	1	-
attn $s(l_4 = 2)$	0	0.17	0.5	0.33	-

Figure 1: An example computation of the attention gates with the LVM. Let $t = 4$.

	0	1	2	3	4
raw attn $\tilde{s}$	0.4	0.1	0.3	0.2	
scores d	0.7	0.6	0.3	0.4	0.5
α	-	0.45	0.6	0.55	-
$p(l_4)$	0.45*0.6*0.55=0.1485	(1-0.45)*0.6*0.55=0.1815	(1-0.6)*0.55=0.22	1-0.55=0.45	
cdf F_{l_4}	0.1485	0.33	0.55	1	
soft attn $s(\mathbb{E}[l_4])$	0.13	0.07	0.36	0.43	-
hard attn $s(l_4 = 2)$	0	0.17	0.5	0.33	-

Figure 2: An example computation of the attention gates. Let $\alpha_i = \frac{d_t - d_i + 1}{2}$ and $t = 4$.

2.2 $p(l_t \mid \mathbf{x}_{\leq t})$

We define

$$p(l_t = k \mid \mathbf{x}_{\leq t}) = \begin{cases} 1 - \alpha_{t-1}, & k = t-1 \\ \prod_{j=1}^{t-1} \alpha_j, & k = 0 \\ (1-\alpha_k) \prod_{j=k+1}^{t-1} \alpha_j, & \text{otherwise} \end{cases} \tag{4}$$

Where the α_k are drawn from an unspecified Beta distribution. Thus $l_t \sim \text{Cat}(\theta)$ where the parameters θ are determined through a stick-breaking process. This ends up looking more like a generalized geometric distribution where probability of success at each time step is flexible and there is a maximum number of trials. This hints at an order-statistic interpretation which we will touch upon if there is time.

Note that this cannot be called a Dirichlet Process, as a $DP(\alpha, H_0)$ is defined by $b_i \sim \text{Beta}(1, \alpha), \pi_i = (1 - b_i) \prod_{j=0}^{i-1} b_i, \boldsymbol{\pi} \sim GEM(\alpha)$ and with $\theta_i \sim H_0$ we have $G = \sum_i \pi_i \delta_{\theta_i} \sim DP(\alpha, H_0)$. This would only make sense in this context if the base distribution H_0 were also Dirichlet, but this would likely warrant further discussion.

Rather than proceed further in defining the generative model, Shen et al. (2018) pivot and use $\mathbb{E}_{l_t}[s_i(l_t)]$ instead of $s_i(l_t)$. We have

$$\mathbb{E}_{l_t}[g_i(l_t)] = \prod_{j=i+1}^{t-1} \alpha_j, \tag{5}$$

and we then use

$$\mathbb{E}_{l_t}[s_i(l_t)] = \frac{\mathbb{E}_{l_t}[g_i(l_t)]\, \tilde{s}_i}{\sum_j \mathbb{E}_{l_t}[g_j(l_t)]\, \tilde{s}_j} \tag{6}$$

as the attention coefficients instead of $s_i(l_t)$. They then parameterize the α_k of timestep t deterministically as follows:

$$\alpha_k = \frac{\text{hardtanh}(2(d_t - d_k)/\tau + 1) + 1}{2}, \tag{7}$$

where τ is some scaling parameter. Any function whose range is $[0, 1]$ should be fine for a stick-breaking definition. The scores $d_k \in \mathbb{R}^+$ are obtained from a CNN over the tokens $\mathbf{x}_{k-L:k}$, where L is the width of the convolution kernel (this is set to 5). The scores d_k are output by the Parsing Network.

[Work through Figure 2]

3 Training

With the soft model that uses $\mathbb{E}[s_i(l_t)]$ training is very simple. We simply maximize the likelihood of the data directly since the model has no latent variables.

4 Extracting a Parse Tree

Given a trained model, we can extract a binary constituency tree using the scores from the parsing network. Perform the following procedure recursively:

```
function SPLIT($x = \{x_0, \dots, x_T\}$)
    if $x = \emptyset$ then
        return $\emptyset$
    else if $|x| = 1$ then
        return $x$
    else
        Choose index $i \in \{0, \dots, T\}$
        return $(\text{SPLIT}(\mathbf{x}_{<i}), (x_i, \text{SPLIT}(\mathbf{x}_{>i})))$
    end if
end function
```

The definition then reduces to defining the index choice step. Under an order-statistic model we could sample $i \sim \text{Cat}(\text{softmax}(\mathbf{d}))$ or use a greedy approximation by taking the argmax of that distribution at every step.

5 Results

5.1 Language Modeling

The language modeling results are decent, but it's unclear whether the gating mechanism offers any improvement. They report the results of an ablation study, but they do not seem to be replicable. On github someone reported that removing the Parsing Network had very little effect on perplexity. It is possible the perplexity improvements come mainly from embedding-tying and partly from self-attention.

SOTA is around 55 and Shen et al. (2018) are around 62.

Model	PPL
RNN-LDA + KN-5 + cache (?)	92.0
LSTM (?)	78.4
Variational LSTM (?)	78.9
CharCNN (?)	78.9
Pointer Sentinel-LSTM (?)	70.9
LSTM + continuous cache pointer (?)	72.1
Variational LSTM (tied) + augmented loss (?)	68.5
Variational RHN (tied) (?)	65.4
NAS Cell (tied) (?)	62.4
4-layer skip connection LSTM (tied) (?)	**58.3**
PRPN	61.98

Table 1: PPL on the Penn Treebank test set

5.2 Parsing

The parsing results, on the other hand, are quite good. On WSJ10, the model beats the right-branching baseline but not the CCM model. On the full WSJ, the model gets an averaged sentence F1 of 38.1, which beats random, balanced, and right branching baselines (the highest F1 reported is 21.3). Why should we care about this? There is no reason apriori that limiting self-attention should work at all for learning to rank, as a fully left-branching structure would allow the self-attention to function as normal.

Model	UF$_1$
LBRANCH	28.7
RANDOM	34.7
DEP-PCFG (**?**)	48.2
RBRANCH	61.7
CCM (**?**)	71.9
DMV+CCM (**?**)	77.6
UML-DOP (**?**)	**82.9**
PRPN	66.4
UPPER BOUND	88.1

Table 2: Parsing Performance on the WSJ10 dataset

5.3 Alternative Probabilistic Interpretation

Rather than defining latent variable l_t which is a categorical over indices, what if we instead define l_t as a function of pairwise comparisons between d_t and all preceeding d_i (for a given timestep t)? We would have

$$l_t = \max\{i : \alpha_i = 1, 0 \leq i < t\}, \alpha_i \sim \text{Bern}\left(\frac{d_i}{d_t + d_i}\right).$$

This yields $p(\alpha_i = 1) = \frac{d_i}{d_t+d_i}, \forall i \in \{1, t-1\}$. Similarly to the stick-breaking procedure, $p(\alpha_i = 0)$ is already defined as the remaining mass. We then have

$$p(l_t = k) = \begin{cases} p(\alpha_{t-1}), & k = t-1 \\ \prod_{j=1}^{t-1} 1 - p(\alpha_j), & k = 0 \\ p(\alpha_k) \prod_{j=k+1}^{t-1} 1 - p(\alpha_j), & \text{otherwise} \end{cases}$$

Again, we would have $l_t \sim$ Cat parameterized as above. However, we could use hard attention tricks to maximize the marginal likelihood of this model exactly and the generative model is actually well-defined. This is known as a Bradley-Terry pairwise order statistic model. See Figure 3 for a worked example of some of this process. Under this model, the implementation of l_t can be interpreted as allowing the model to attend up through the nearest token to the left that has higher rank. This has a tree interpretation, but it is unintuitive. Once we learn a pairwise model, we can use the parameterization of the scores d_i to define a model over trees. The tree-based interpretation would then be:

(a) either 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.

To define a distribution over trees using scores, we follow a generative story similar to the split function. In order to define a tree, we recursively partition $\mathbf{x}$:

$$p(\mathbf{x}; \mathbf{d}) = p(\text{choose} = i; \mathbf{d}) p(\mathbf{x}_{<t}; \mathbf{d}_{<t}) p(\mathbf{x}_{>t}; \mathbf{d}_{>t}), \tag{8}$$

where $p(\text{choose} = i; \mathbf{d}) = \frac{d_i}{\sum_j d_j}$.

If we follow the split function exactly, we would instead define a distribution over rankings:

$$p(\mathbf{x}; \mathbf{d}) = p(\text{choose} = i; \mathbf{d}) p(\mathbf{x} \setminus \{x_i\}; \mathbf{d} \setminus \{d_i\}), \tag{9}$$

where the choice is parameterized in exactly the same way. Under this model, the split procedure can be seen as a greedy approach to finding the most likely tree under this distribution.

References

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`.

	0	1	2	3	4
scores d	0.7	0.6	0.3	0.4	0.5
$p(\alpha_i = 1)$	-	6/11	3/8	4/9	-
$p(l_4 = k)$	0.16	0.19	(1-0.45)*3/8= 0.21	4/9=0.45	-

Figure 3: An example computation of the attention gates. with a pairwise ranking model and $t = 4$. It gives very similar probabilities when compared to the 'stick-breaking' approach but is tractable.