Paper Reading: Ordered Neurons: Integrating Tree Structures into Recurrent Neural Networks

Yikang Shen, Shawn Tan, Alessandro Sordoni, Aaron Courville

MILA/University of Montreal

Motivation

- Integrating tree structure into a language model is important:
  - to obtain a hierarchical representation with increasing levels of abstraction, a key feature of deep neural networks
  - to capture complex linguistic phenomena, like the long-term dependency problem and the compositional effects
  - to provide shortcut for gradient back-propagation
- Previous work shows that RNN models could learn to exploit the underlying tree structure. However, its performance consistently lags behind that of tree-based models.
- Propose a new inductive bias: Ordered Neurons: some high-ranking neurons store long-term information, while low-ranking neurons store short-term information.

The relationship between a constituency parse tree and an ON-LSTM

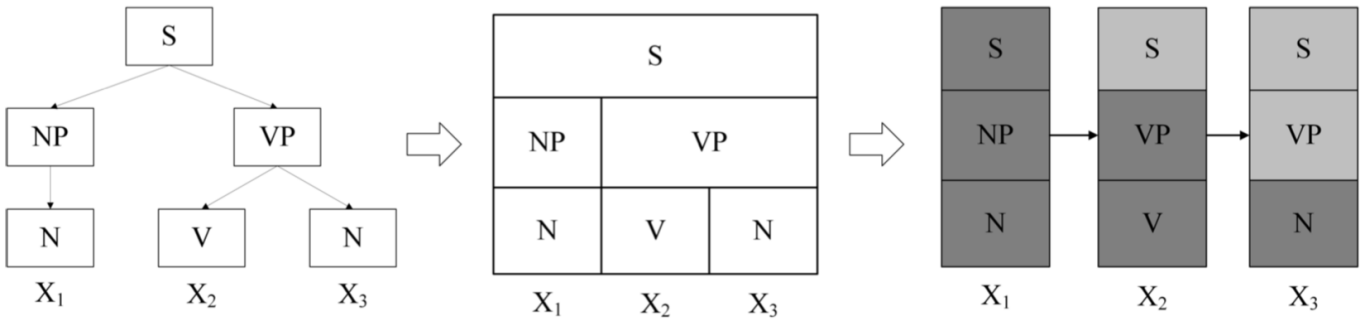

Figure 1: The relationship between a constituency parse tree and an ON-LSTM. Given a sequence of tokens (x_1, x_2, x_3), their constituency-based parse tree is illustrated in (a). (b) provides a block view of the tree structure, where S and VP node strides across more then one time step. The representation for high-ranking nodes should be relatively consistent across multiple time steps. (c) visualization of the ratio of updated neurons for each group of neurons at each time step. At each time step, given the input word, darker grey blocks are completely updated, lighter grey blocks are partially updated. The three groups of neurons have different update frequencies. Higher groups update less frequently and lower groups update more frequently.

Ordered Neurons

Assumptions:

- A order should exist between neurons: the high-ranking neurons store long-term information, while the low-ranking neurons store short-term information. To erase(or update) high-ranking neurons, the model should first erase(or update) all lower-ranking neurons.
- This ordering is independent of the data, thus we can enforce it on hidden states as an inductive bias.

Implementation of ordered neurons: ON-LSTM

$$
\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
\hat{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
$$

The only difference with the standard LSTM is that excluding the update function for cell state c_t and replace it with a new update rule.

Activation Function: cumax()

$\hat{g} = cumax(...) = cumsum(softmax(...))$

The binary gate split the cell state into two segments: the 0-segment and the 1-segment. The index for the first 1 in g is parametrised as: $p(d) = softmax(...)$. The discrete variable d represents the split point between the two segments.

Cumulative distribution function:
$p(g_k = 1) = p(d <= k) = \sum_{i<=k} p(d = i)$.

Structured Gating Mechanism

Master forget gate and master input gate:

$$\tilde{f}_t = \text{cumax}(W_{\tilde{f}} x_t + U_{\tilde{f}} h_{t-1} + b_{\tilde{f}})$$
$$\tilde{i}_t = 1 - \text{cumax}(W_{\tilde{i}} x_t + U_{\tilde{i}} h_{t-1} + b_{\tilde{i}})$$

New update rule:

$$\omega_t = \tilde{f}_t \circ \tilde{i}_t \tag{11}$$
$$\hat{f}_t = f_t \circ \omega_t + (\tilde{f}_t - \omega_t) = \tilde{f}_t \circ (f_t \circ \tilde{i}_t + 1 - \tilde{i}_t) \tag{12}$$
$$\hat{i}_t = i_t \circ \omega_t + (\tilde{i}_t - \omega_t) = \tilde{i}_t \circ (i_t \circ \tilde{f}_t + 1 - \tilde{f}_t) \tag{13}$$
$$c_t = \hat{f}_t \circ c_{t-1} + \hat{i}_t \circ \hat{c}_t \tag{14}$$

Intuition behind the new update rule:

- The master forget gate $\widetilde{f}$ controls the erasing behavior of the model. Suppose $\widetilde{f}_t = (0, ..., 0, 1, ..., 1)$ and the split point is d_t^f . Given the Eq. (12) and (14), the information stored in the first d_t^f neurons of the previous cell states $c_{t?1}$ will be completely erased.
- The master input gate $\widetilde{i}_t$ is meant to control the writing behavior of model. A large d_t^f means that the current input xt contains long-term information that needs to be preserved for several time steps.
- The product of two master gates w_t represents the overlap of $\widetilde{f}_t$ and $\widetilde{i}_t$. For example, in figure 1, the word x_3 belongs to the constituents S and VP . At this time step, the overlap w_3 would cover the related blocks of neurons, such that these neurons could be partial updated.

Experiments: Language Modeling

Model	Parameters	Validation	Test
Zaremba et al. (2014) - LSTM (large)	66M	82.2	78.4
Gal & Ghahramani (2016) - Variational LSTM (large, MC)	66M	–	73.4
Kim et al. (2016) - CharCNN	19M	–	78.9
Merity et al. (2016) - Pointer Sentinel-LSTM	21M	72.4	70.9
Grave et al. (2016) - LSTM	–	–	82.3
Grave et al. (2016) - LSTM + continuous cache pointer	–	–	72.1
Inan et al. (2016) - Variational LSTM (tied) + augmented loss	51M	71.1	68.5
Zilly et al. (2016) - Variational RHN (tied)	23M	67.9	65.4
Zoph & Le (2016) - NAS Cell (tied)	54M	–	62.4
Shen et al. (2017) - PRPN-LM	–	–	62.0
Melis et al. (2017) - 4-layer skip connection LSTM (tied)	24M	60.9	58.3
Merity et al. (2017) - AWD-LSTM - 3-layer LSTM (tied)	24M	60.0	57.3
ON-LSTM - 3-layer (tied)	25M	$\mathbf{58.29 \pm 0.10}$	$\mathbf{56.17 \pm 0.12}$
Yang et al. (2017) - AWD-LSTM-MoS*	22M	56.5	54.4

Table 1: Single model perplexity on validation and test sets for the Penn Treebank language modeling task. Models noting *tied* use weight tying on the embedding and softmax weights. Model noting * focus on improving the softmax component of RNN language model. Their contribution is orthogonal to ours.

Experiments: Unsupervised Constituency Parsing

Model	Training Data	Training Object	Vocab Size	Parsing F1 WSJ10 μ (σ)	WSJ10 max	WSJ μ (σ)	WSJ max	Depth WSJ	Accuracy on WSJ by Tag ADJP	NP	PP	INTJ
PRPN-UP	AllNLI Train	LM	76k	66.3 (0.8)	68.5	38.3 (0.5)	39.8	5.8	28.7	65.5	32.7	*0.0*
PRPN-LM	AllNLI Train	LM	76k	52.4 (4.9)	58.1	35.0 (5.4)	42.8	6.1	37.8	59.7	**61.5**	**100.0**
PRPN-UP	WSJ Train	LM	15.8k	62.2 (3.9)	70.3	26.0 (2.3)	32.8	5.8	24.8	54.4	17.8	*0.0*
PRPN-LM	WSJ Train	LM	10k	70.5 (0.4)	71.3	37.4 (0.3)	38.1	5.9	26.2	**63.9**	24.4	*0.0*
ON-LSTM 1st-layer	WSJ Train	LM	10k	35.2(4.1)	*42.8*	20.0(2.8)	24.0	5.6	38.1	23.8	18.3	**100.0**
ON-LSTM 2nd-layer	WSJ Train	LM	10k	65.1(1.7)	66.8	47.7(1.5)	**49.4**	5.6	**46.2**	61.4	55.4	*0.0*
ON-LSTM 3rd-layer	WSJ Train	LM	10k	54.0(3.9)	57.6	36.6(3.3)	40.4	5.3	44.8	57.5	47.2	*0.0*
300D ST-Gumbel	AllNLI Train	NLI	–	–	–	*19.0 (1.0)*	*20.1*	–	*15.6*	*18.8*	*9.9*	59.4
w/o Leaf GRU	AllNLI Train	NLI	–	–	–	22.8 (1.6)	25.0	–	18.9	24.1	*14.2*	51.8
300D RL-SPINN	AllNLI Train	NLI	–	–	–	*13.2 (0.0)*	*13.2*	–	*1.7*	*10.8*	*4.6*	50.6
w/o Leaf GRU	AllNLI Train	NLI	–	–	–	*13.1 (0.1)*	*13.2*	–	*1.6*	*10.9*	*4.6*	50.0
CCM	WSJ10 Full	–	–	–	71.9	–	–	–	–	–	–	–
DMV+CCM	WSJ10 Full	–	–	–	77.6	–	–	–	–	–	–	–
UML-DOP	WSJ10 Full	–	–	–	**82.9**	–	–	–	–	–	–	–
Random Trees	–	–	–	–	34.7	21.3 (0.0)	21.4	5.3	17.4	22.3	16.0	40.4
Balanced Trees	–	–	–	–	–	21.3 (0.0)	21.3	4.6	22.1	*20.2*	*9.3*	55.9
Left Branching	–	–	–	*28.7*	28.7	13.1 (0.0)	13.1	12.4	–	–	–	–
Right Branching	–	–	–	61.7	61.7	16.5 (0.0)	16.5	12.4	–	–	–	–

Table 2: Unlabeled parsing F1 results evaluated on full WSJ10 and WSJ test set. Our language model has three layers, each of them provides a sequence of $\hat{d}_t$. We provide the parsing performance for all layers. Results with RL-SPINN and ST-Gumbel are evaluated on the full WSJ (Williams et al., 2017). PRPN models are evaluated on WSJ test set (Htut et al., 2018). We run the model with 5 different random seeds to calculate the average F1. The *Accuracy* columns represent the fraction of ground truth constituents of a given type that corresponds to constituents in the model parses. We use the model with the best F1 score to report ADJP, NP, PP, and INTJ. WSJ10 baselines are from Klein & Manning (2002, CCM), Klein & Manning (2005, DMV+CCM), and Bod (2006, UML-DOP). As the WSJ10 baselines are trained using additional information such as POS tags and dependency parser, they are not strictly comparable with the latent tree learning results. Italics mark results that are worse than the random baseline.

Experiments: Targeted Syntactic Evaluation

	ON-LSTM	LSTM
Short-Term Dependency		
SUBJECT-VERB AGREEMENT:		
Simple	0.99	**1.00**
In a sentential complement	0.95	**0.98**
Short VP coordination	0.89	**0.92**
In an object relative clause	0.84	**0.88**
In an object relative (no *that*)	0.78	**0.81**
REFLEXIVE ANAPHORA:		
Simple	**0.89**	0.82
In a sentential complement	**0.86**	0.80
NEGATIVE POLARITY ITEMS:		
Simple (grammatical vs. intrusive)	0.18	**1.00**
Simple (intrusive vs. ungrammatical)	**0.50**	0.01
Simple (grammatical vs. ungrammatical)	0.07	**0.63**
Long-Term Dependency		
SUBJECT-VERB AGREEMENT:		
Long VP coordination	**0.74**	**0.74**
Across a prepositional phrase	0.67	**0.68**
Across a subject relative clause	**0.66**	0.60
Across an object relative clause	**0.57**	0.52
Across an object relative (no *that*)	**0.54**	0.51
REFLEXIVE ANAPHORA:		
Across a relative clause	0.57	**0.58**
NEGATIVE POLARITY ITEMS:		
Across a relative clause (grammatical vs. intrusive)	0.59	**0.95**
Across a relative clause (intrusive vs. ungrammatical)	**0.20**	0.00
Across a relative clause (grammatical vs. ungrammatical)	**0.11**	0.04

Table 3: Overall accuracy for the ON-LSTM and LSTM on each test case. "Long-term dependency" means that an unrelated phrase (or a clause) exist between the targeted pair of words, while "short-term dependency" means there is no such distraction.

Experiments: Logical Inference

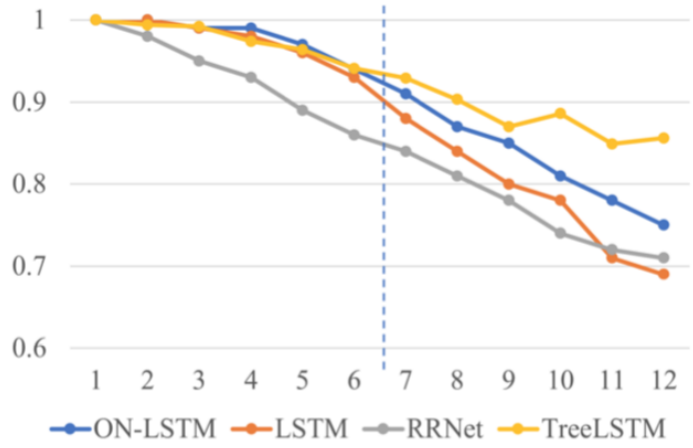

Figure 2: Test accuracy of the models, trained on short sequences (≤ 6) in logic data. The horizontal axis indicates the length of the sequence, and the vertical axis indicates the accuracy of models performance on the corresponding test set.

Conclusion

- The paper introduces an elegant way of adding a hierarchical inductive bias; the intuition behind this idea is explained clearly.
- RNNs have power to model tree structure.