Supplementary Material for "Learning to Compose Task-Specific Tree Structures"

Jihun Choi, Kang Min Yoo, Sang-goo Lee
Seoul National University, Seoul 08826, Korea
{jhchoi, kangminyoo, sglee}@europa.snu.ac.kr

Implementation Details

Implementation-wise, we used multiple *mask* matrices in implementing the proposed Gumbel Tree-LSTM model. Using the mask matrices, Eq. 11 can be rewritten as a single equation:

$$\mathbf{r}^{t+1}_{1:M_{t+1}} = \mathbf{M}_l \odot \mathbf{r}^t_{1:M_t-1} + \mathbf{M}_r \odot \mathbf{r}^t_{2:M_t} + \mathbf{M}_p \odot \tilde{\mathbf{r}}^{t+1}_{1:M_{t+1}}. \tag{S1}$$

In the above equation, $\mathbf{M}_l, \mathbf{M}_r, \mathbf{M}_p \in \mathbb{R}^{D_h \times M_{t+1}}$, and $\mathbf{r}^t_{1:L} \in \mathbb{R}^{D_h \times L}$ is a matrix whose columns are $\mathbf{r}^t_1, \cdots, \mathbf{r}^t_L \in \mathbb{R}^{D_h}$.

The mask matrices are defined by the following equations.

$$\mathbf{M}_l = [\mathbf{m}_l \quad \cdots \quad \mathbf{m}_l]^T \tag{S2}$$

$$\mathbf{M}_r = [\mathbf{m}_r \quad \cdots \quad \mathbf{m}_r]^T \tag{S3}$$

$$\mathbf{M}_p = [\mathbf{m}_p \quad \cdots \quad \mathbf{m}_p]^T \tag{S4}$$

$$\mathbf{m}_l = \mathbf{1} - \texttt{cumsum}(\bar{\mathbf{y}}_{1:M_{t+1}}) \tag{S5}$$

$$\mathbf{m}_r = [0 \quad \texttt{cumsum}(\bar{\mathbf{y}}_{1:M_{t+1}-1})]^T \tag{S6}$$

$$\mathbf{m}_p = \bar{\mathbf{y}}_{1:M_{t+1}} \tag{S7}$$

Here, $\texttt{cumsum}(\mathbf{c})$ is a function that takes a vector $\mathbf{c} = [c_1 \quad \cdots \quad c_k]^T$ and outputs a vector $\mathbf{d} = [d_1 \quad \cdots \quad d_k]^T$ s.t. $d_i = \sum_{j=1}^{i} c_j$. $\bar{\mathbf{y}}_{1:M_{t+1}} \in \mathbb{R}^{M_{t+1}}$ is a vector which will be defined below, and $\mathbf{1} \in \mathbb{R}^{M_{t+1}}$ is a vector whose values are all ones.

In the forward pass, $\bar{\mathbf{y}}_{1:M_{t+1}}$ is defined by a one-hot vector $\mathbf{y}^{ST}_{1:M_{t+1}}$, which is sampled from the categorical distribution of validity scores $(v_1, \cdots, v_{M_{t+1}})$ using Gumbel-Max trick.

$$y^{ST}_i = \begin{cases} 1 & i = \arg\max_j \left(\mathbf{q} \cdot \tilde{\mathbf{h}}^{t+1}_j + g_j \right) \\ 0 & \text{otherwise} \end{cases} \tag{S8}$$

$$g_i = -\log(-\log(u_i + \epsilon) + \epsilon) \tag{S9}$$

$$u_i \sim \text{Uniform}(0, 1) \tag{S10}$$

Note that $\epsilon = 10^{-20}$ is added when calculating g_i for numerical stability.

In the backward pass, instead of the one-hot version, the continuous vector $\mathbf{y}_{1:M_{t+1}}$ obtained from Gumbel-Softmax is used as $\bar{\mathbf{y}}_{1:M_{t+1}}$. Note that the Gumbel noise samples $g_1, \cdots, g_{M_{t+1}}$ drawn in the forward pass are reused in the backward pass (i.e. noise values are not resampled in the backward pass).

In typical deep learning libraries supporting automatic differentiation (e.g. PyTorch, TensorFlow), this discrepancy between forward and backward pass can be implemented as

$$\bar{\mathbf{y}}_{1:M_{t+1}} = \texttt{detach}(\mathbf{y}^{ST}_{1:M_{t+1}} - \mathbf{y}_{1:M_{t+1}}) + \mathbf{y}_{1:M_{t+1}}, \tag{S11}$$

where $\texttt{detach}(\cdot)$ is a function that prevents error from back-propagating through its input.

Detailed Experimental Settings

All experiments are conducted using the publicized codebase.[1]

SNLI

The composition query vector is initialized by sampling from Gaussian distribution $\mathcal{N}(0, 0.01^2)$. The last linear transformation that outputs the unnormalized log probability for each class is initialized by sampling from uniform distribution $\mathcal{U}(-0.005, 0.005)$. All other parameters are initialized following the scheme proposed by He et al. (2015). We used Adam optimizer (Kingma and Ba 2015) with default hyperparameters and halved learning rate if there is no improvement in accuracy for one epoch. The size of mini-batch is set to 128 in all experiments.

In 100D experiments ($D_x = D_h = 100$, $D_c = 200$, single-hidden layer MLP classifier), GloVe (6B, 100D) (Pennington, Socher, and Manning 2014) pretrained word embeddings are used in initializing word representations. We fine-tuned word embedding parameters during training.

In 300D ($D_x = D_h = 300$, $D_c = 1024$, single-hidden layer MLP classifier) and 600D ($D_x = 300$, $D_h = 600$, $D_c = 1024$, MLP classifier with three hidden layers) experiments, GloVe (840B, 300D) pretrained word embeddings are used as word representations and fixed during training. Batch normalization is applied before the input and after the output of the MLP. Dropout is applied to word embeddings and the input and the output of the MLP with dropout probability 0.1 (300D) or 0.2 (600D).

[1] https://github.com/jihunchoi/unsupervised-treelstm

SST

The composition query vector is initialized by sampling from Gaussian distribution $\mathcal{N}(0, 0.01^2)$. The last linear transformation that outputs the unnormalized log probability for each class is initialized by sampling from uniform distribution $\mathcal{U}(-0.002, 0.002)$. All other parameters are initialized following the scheme proposed by He et al. (2015). We used Adadelta optimizer (Zeiler 2012) with default hyperparameters and halved learning rate if there is no improvement in accuracy for two epochs. In both SST-2 and SST-5 experiments, we set $D_x = D_h = 300$, used GloVe (840B, 300D) pretrained vectors with fine-tuning, and single-hidden layer MLP is used as classifier. Dropout is applied to word embeddings and the input and the output of the MLP classifier with probability 0.5.

In the SST-2 experiment, we set D_c to 300 and set batch size to 32. In the SST-5 experiment, D_c is increased to 1024, and mini-batches of 64 sentences are fed to the model during training.

References

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, 1026–1034.

Kingma, D. P., and Ba, J. L. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. GloVe: Global vectors for word representation. In *EMNLP*, 1532–1543.

Zeiler, M. D. 2012. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.