Supplementary Material for "Learning to Compose Dynamic Tree Structures for Visual Contexts"

Kaihua Tang[1], Hanwang Zhang[1], Baoyuan Wu[2], Wenhan Luo[2], Wei Liu[2]
[1] Nanyang Technological University
[2] Tencent AI Lab
kaihua001@e.ntu.edu.sg, *hanwangzhang@ntu.edu.sg*,
{*wubaoyuan1987, whluo.china*}*@gmail.com*, *wl2223@columbia.edu*

1. Bidirectional TreeLSTM

In this section, we will introduce the details of the bidirectional TreeLSTM applied to encode the object-level visual contexts. For the bottom-up direction, we employ N-ary TreeLSTM [2] for binary trees, *i.e.*, VCTREEs and Overlap Trees, and the normalized Child-Sum [2] TreeLSTM for Multi-Branch Trees. For the top-down direction, since each node only has one parent, TreeLSTM is similar to the traditional LSTM [1].

1.1. N-ary TreeLSTM for Binary Trees

According to the definition of N-ary TreeLSTM [2], it can be applied to the tree structures with at most N ordered branches for each node. In our work, we adopt binary TreeLSTM as our bottom-up TreeLSTM for the proposed binary tree structures, *i.e.*, VCTREEs and Overlap Trees. It can be formulated as follows:

$$\overleftarrow{\boldsymbol{h}}_t = \text{TreeLSTM}\big(\boldsymbol{z}_t, [\overleftarrow{\boldsymbol{h}}_l; \overleftarrow{\boldsymbol{h}}_r]\big), \tag{1}$$

$$\boldsymbol{i}_t = \sigma\big(\boldsymbol{W}^{(i)}\boldsymbol{z}_t + \boldsymbol{U}^{(i)}[\overleftarrow{\boldsymbol{h}}_l; \overleftarrow{\boldsymbol{h}}_r] + \boldsymbol{b}^{(i)}\big), \tag{2}$$

$$\boldsymbol{f}_l = \sigma\big(\boldsymbol{W}_l^{(f)}\boldsymbol{z}_t + \boldsymbol{U}_l^{(f)}[\overleftarrow{\boldsymbol{h}}_l; \overleftarrow{\boldsymbol{h}}_r] + \boldsymbol{b}_l^{(f)}\big), \tag{3}$$

$$\boldsymbol{f}_r = \sigma\big(\boldsymbol{W}_r^{(f)}\boldsymbol{z}_t + \boldsymbol{U}_r^{(f)}[\overleftarrow{\boldsymbol{h}}_l; \overleftarrow{\boldsymbol{h}}_r] + \boldsymbol{b}_r^{(f)}\big), \tag{4}$$

$$\boldsymbol{o}_t = \sigma\big(\boldsymbol{W}^{(o)}\boldsymbol{z}_t + \boldsymbol{U}^{(o)}[\overleftarrow{\boldsymbol{h}}_l; \overleftarrow{\boldsymbol{h}}_r] + \boldsymbol{b}^{(o)}\big), \tag{5}$$

$$\boldsymbol{u}_t = \tanh\big(\boldsymbol{W}^{(u)}\boldsymbol{z}_t + \boldsymbol{U}^{(u)}[\overleftarrow{\boldsymbol{h}}_l; \overleftarrow{\boldsymbol{h}}_r] + \boldsymbol{b}^{(u)}\big), \tag{6}$$

$$\overleftarrow{\boldsymbol{c}}_t = \boldsymbol{i}_t \odot \boldsymbol{u}_t + \boldsymbol{f}_l \odot \overleftarrow{\boldsymbol{c}}_l + \boldsymbol{f}_r \odot \overleftarrow{\boldsymbol{c}}_r, \tag{7}$$

$$\overleftarrow{\boldsymbol{h}}_t = \boldsymbol{o}_t \odot \tanh(\overleftarrow{\boldsymbol{c}}_t), \tag{8}$$

where $\boldsymbol{z}_t \in \mathbb{R}^d$ is the input feature for node t; $\overleftarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_l, \overleftarrow{\boldsymbol{h}}_r \in \mathbb{R}^h$ are the hidden states; $\overleftarrow{\boldsymbol{c}}_t, \overleftarrow{\boldsymbol{c}}_l, \overleftarrow{\boldsymbol{c}}_r \in \mathbb{R}^h$ are memory cells; $\boldsymbol{W}^{(i)}, \boldsymbol{W}_l^{(f)}, \boldsymbol{W}_r^{(f)}, \boldsymbol{W}^{(o)}, \boldsymbol{W}^{(u)} \in \mathbb{R}^{h\times d}$ and $\boldsymbol{U}^{(i)}, \boldsymbol{U}_l^{(f)}, \boldsymbol{U}_r^{(f)}, \boldsymbol{U}^{(o)}, \boldsymbol{U}^{(u)} \in \mathbb{R}^{h\times 2h}$ are learnable matrices; $\boldsymbol{b}^{(i)}, \boldsymbol{b}_l^{(f)}, \boldsymbol{b}_r^{(f)}, \boldsymbol{b}^{(o)}, \boldsymbol{b}^{(u)} \in \mathbb{R}^h$ are vectors; σ denotes sigmoid function; tanh denotes tanh activation function; $\odot$ means element-wise product. Note that we slightly abuse the subscripts l, r of $\overleftarrow{\boldsymbol{c}}_l, \overleftarrow{\boldsymbol{c}}_r, \overleftarrow{\boldsymbol{h}}_l, \overleftarrow{\boldsymbol{h}}_r$ to denote hidden states and memory cells from the left-child and right-child of node t. The hidden states and memory cells of the missing branches will be filled with zero vectors.

1.2. Child-Sum TreeLSTM for Multi-Branch Trees

The Child-Sum TreeLSTM [2] is able to deal with the tree structure where each node has arbitrary number of children. Therefore, we adopt it as the bottom-up TreeLSTM of the context encoder for the Multi-Branch Trees in the ablation studies. For each node t of a Multi-Branch Tree, we define $C(t)$ as the set of its children. Compared with the original paper [2], we replace the Child-Sum with the Child-Mean in our implementation for better normalization, then it is formulated as:

$$\overleftarrow{\boldsymbol{h}}_t = \text{TreeLSTM}\big(\boldsymbol{z}_t, \{\overleftarrow{\boldsymbol{h}}_k\}\big), k \in C(t), \tag{9}$$

$$\overleftarrow{\boldsymbol{h}}_{mean} = \frac{\sum_{k\in C(t)} \overleftarrow{\boldsymbol{h}}_k}{|C(t)|}, \tag{10}$$

$$\boldsymbol{i}_t = \sigma\big(\boldsymbol{W}^{(i)}\boldsymbol{z}_t + \boldsymbol{U}^{(i)}\overleftarrow{\boldsymbol{h}}_{mean} + \boldsymbol{b}^{(i)}\big), \tag{11}$$

$$\boldsymbol{f}_k = \sigma\big(\boldsymbol{W}^{(f)}\boldsymbol{z}_t + \boldsymbol{U}^{(f)}\overleftarrow{\boldsymbol{h}}_k + \boldsymbol{b}^{(f)}\big), \tag{12}$$

$$\boldsymbol{o}_t = \sigma\big(\boldsymbol{W}^{(o)}\boldsymbol{z}_t + \boldsymbol{U}^{(o)}\overleftarrow{\boldsymbol{h}}_{mean} + \boldsymbol{b}^{(o)}\big), \tag{13}$$

$$\boldsymbol{u}_t = \tanh\big(\boldsymbol{W}^{(u)}\boldsymbol{z}_t + \boldsymbol{U}^{(u)}\overleftarrow{\boldsymbol{h}}_{mean} + \boldsymbol{b}^{(u)}\big), \tag{14}$$

$$\overleftarrow{\boldsymbol{c}}_t = \boldsymbol{i}_t \odot \boldsymbol{u}_t + \frac{\sum_{k\in C(t)} \boldsymbol{f}_k \odot \overleftarrow{\boldsymbol{c}}_k}{|C(t)|}, \tag{15}$$

$$\overleftarrow{\boldsymbol{h}}_t = \boldsymbol{o}_t \odot \tanh(\overleftarrow{\boldsymbol{c}}_t), \tag{16}$$

where $\overleftarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_k, \in \mathbb{R}^h$ are the hidden states; $\overleftarrow{\boldsymbol{c}}_t, \overleftarrow{\boldsymbol{c}}_k \in \mathbb{R}^h$ are memory cells; $\boldsymbol{W}^{(i)}, \boldsymbol{W}^{(f)}, \boldsymbol{W}^{(o)}, \boldsymbol{W}^{(u)} \in \mathbb{R}^{h\times d}$ and $\boldsymbol{U}^{(i)}, \boldsymbol{U}^{(f)}, \boldsymbol{U}^{(o)}, \boldsymbol{U}^{(u)} \in \mathbb{R}^{h\times h}$ are learnable matrices; $\boldsymbol{b}^{(i)}, \boldsymbol{b}^{(f)}, \boldsymbol{b}^{(o)}, \boldsymbol{b}^{(u)} \in \mathbb{R}^h$ are vectors; $|C(t)|$ is the number of children for node t; $\overleftarrow{\boldsymbol{h}}_{mean}$ denotes the mean hidden state of all the children of node t.

	Scene Graph Generation			Scene Graph Classification			Predicate Classification		
Model	mR@20	mR@50	mR@100	mR@20	mR@50	mR@100	mR@20	mR@50	mR@100
MOTIFS [3]	4.2	5.7	6.6	6.3	7.7	8.2	10.8	14.0	15.3
FREQ [3]	4.5	6.1	7.1	5.1	7.2	8.5	8.3	13.0	16.0
Chain	4.6	6.3	7.2	6.3	7.9	8.8	11.0	14.4	16.6
Overlap	4.8	6.5	7.5	7.2	9.0	9.3	12.5	16.1	17.4
Multi-Branch	4.7	6.5	7.4	6.9	8.6	9.2	11.9	15.5	16.9
VCTREE-SL	5.0	6.7	7.7	8.0	9.8	10.5	13.4	17.0	18.5
VCTREE-HL	**5.2**	**6.9**	**8.0**	**8.2**	**10.1**	**10.8**	**14.0**	**17.9**	**19.4**

Table 1. Mean recall (%) of various methods across all the 50 predicate categories. MOTIFS [3] and FREQ [3] are using the same Faster-RCNN detector as ours.

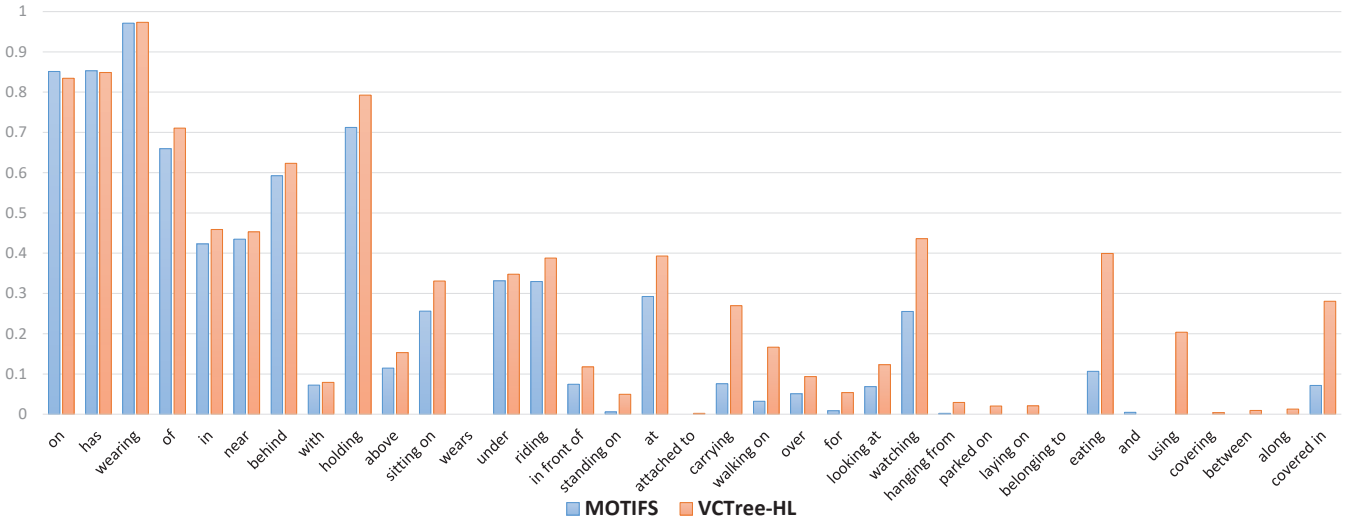

Figure 1. Recall@100 of MOTIFS [3] and the proposed VCTREE-HL under PredCls for each Top-35 category ranking by frequency.

1.3. Top-Down TreeLSTM

We use the traditional LSTM [1] as the top-down TreeLSTM for all the VCTREEs, Overlap Trees, and Multi-Branch Trees, because each node only has at most one parent. The only difference with the traditional LSTM is that our structures are trees rather than chains, the previous hidden state is from the parent of node t.

For the proposed VCTREE, we assigned different learnable matrices for the hidden states from the left-branch parents and right-branch parents. However, the result didn't show significant improvements in the end-tasks, so we employ traditional LSTM as our top-down LSTM for efficiency.

2. Quantitative Analysis

2.1. Mean Recall for Scene Graph

We also report more detailed results of the proposed **Mean Recall (mR@K)** in Table 1. The proposed VCTREE-HL shows best performance among all the ablative structures. Note that MOTIFS [3] has lower mR@100 than FREQ [3] baseline in SGCls and PredCls, which means that MOTIFS is even worse at predicting infrequent predicate categories. However, its mR@20 and mR@50 are higher than FREQ in SGCls and PredCls, which indicates that MOTIFS better separates the foreground relationships from the background ones than FREQ.

2.2. Predicate Recall Analysis

To better visualize the improvement of the proposed VCTREE-HL on infrequent predicate categories, we rank all the predicate categories by frequency, and show the PredCls Recall@100 of MOTIFS [3] and VCTREE-HL for each top-35 category independently in Figure 1. We can observe significant improvements on those less frequent but more semantically meaningful predicates.

3. Qualitative Analysis

3.1. Scene Graph Generation

We further investigated more misclassified results of the proposed VCTREE-HL. The corresponding tree structures and the generated scene graphs are reported in Figure 2. We observed 3 types of interesting misclassifications: 1) In the image (a) of Figure 2, the proposed VCTREE-HL predicts more appropriate predicates "in front of" and "behind" than

original "near". 2) In the image (b) and (d), the ground truth "man in snow" and "window near building" are improper, while our method shows more appropriate predicates. 3) In the image (c) and (d), the objects isolated from the Scene Graph (only considering R@20 predicates) are easier to be misclassified.

3.2. Visual Question Answering

More constructed VCTREEs for VQA2.0 are visualized in Figure 3. The dynamic tree structures are subject to different questions, which allow the objects in an image to incorporate the different contextual cues according to each question. The proposed VCTREE also helps us understand how the model predicts the answer of the question given the image, *e.g.*, in image (a) of Figure 3, given the question "does this dog have a collar?", we find that our model first focuses on the collar-like object rather than the dog; in image (b) of Figure 3, given the question "what sport is being played?", we find that our model focuses on the sportsman rather than playground to answer this question.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997. 1, 2

[2] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, 2015. 1

[3] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *CVPR*, 2018. 2

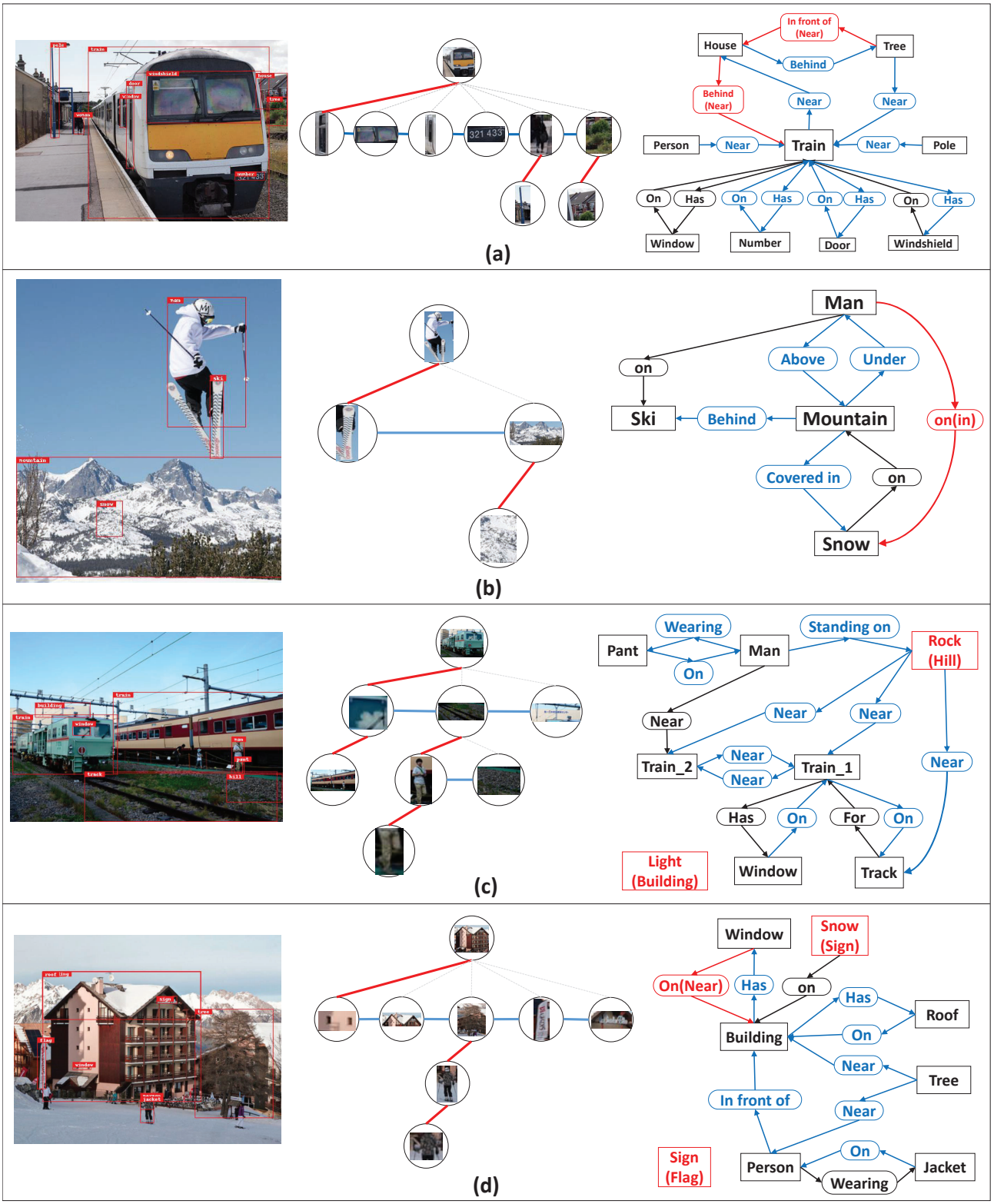

Figure 2. The learned tree structures and generated scene graphs in VG. We selectively report the predicates from R@20 and all the ground-truth predicates. Black color indicates correctly detected objects or predicates; red indicates the misclassified ones; blue indicates correct predictions that not labeled as ground-truth.

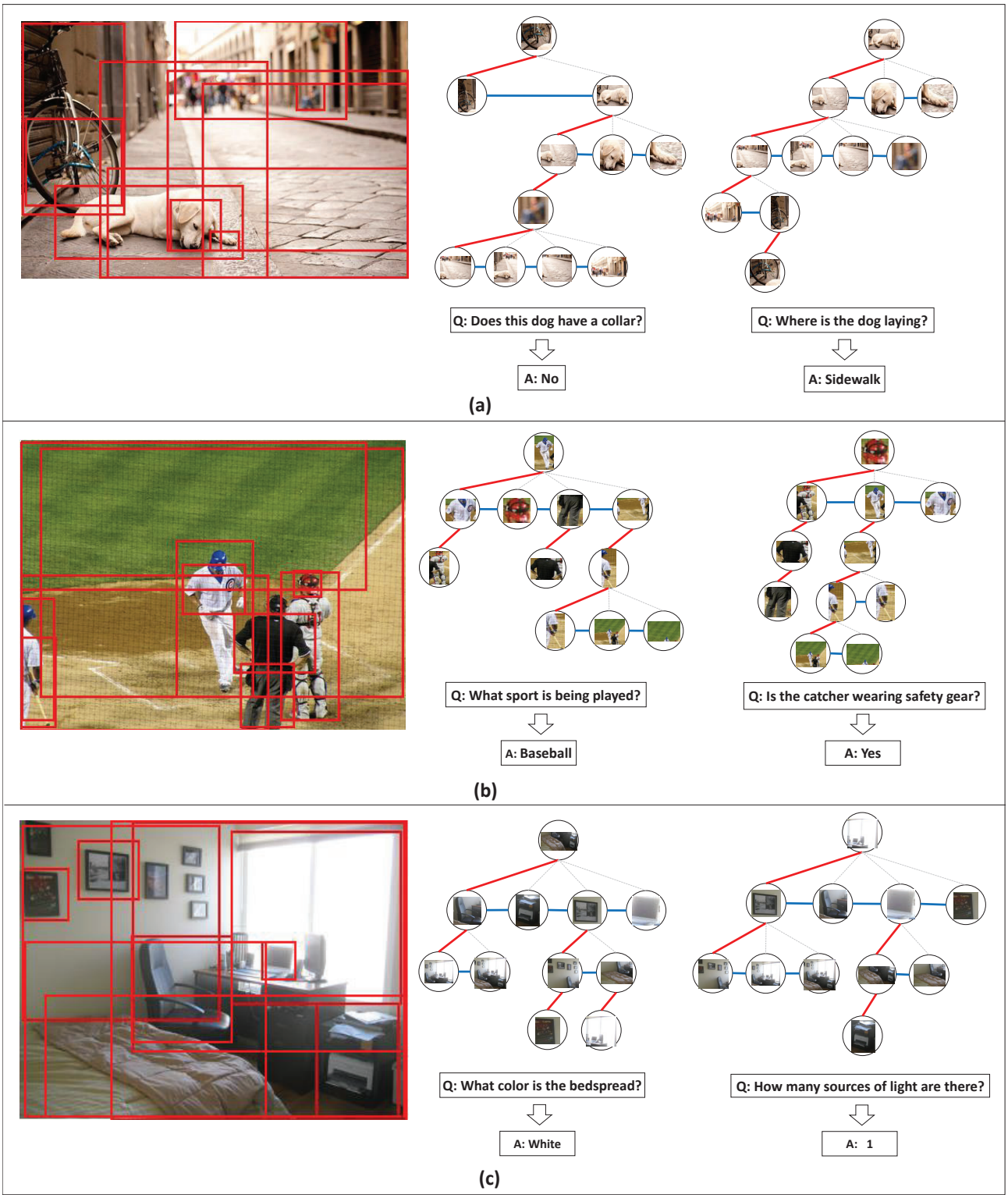

Figure 3. The dynamic and interpretable tree structures that subject to different questions, which allow the objects in an image incorporate different contextual cues according to each question.