Learning to Compose and Reason with Language Tree Structures for Visual Grounding

Richang Hong, Daqing Liu, Xiaoyu Mo, Xiangnan He, Hanwang Zhang

Abstract—Grounding natural language in images, such as localizing "the black dog on the left of the tree", is one of the core problems in artificial intelligence, as it needs to comprehend the fine-grained and compositional language space. However, existing solutions rely on the association between the holistic language features and visual features, while neglect the nature of compositional reasoning implied in the language. In this paper, we propose a natural language grounding model that can automatically compose a binary tree structure for parsing the language and then perform visual reasoning along the tree in a bottom-up fashion. We call our model RvG-Tree: Recursive Grounding Tree, which is inspired by the intuition that any language expression can be recursively decomposed into two constituent parts, and the grounding confidence score can be recursively accumulated by calculating their grounding scores returned by sub-trees. RvG-Tree can be trained end-to-end by using the Straight-Through Gumbel-Softmax estimator that allows the gradients from the continuous score functions passing through the discrete tree construction. Experiments on several benchmarks show that our model achieves the state-of-the-art performance with more explainable reasoning.

Index Terms—Fine-grained detection, tree structure, visual grounding, visual reasoning

1 Introduction

With the maturity of deep neural networks for object detection [1], we are more ambitious to fulfill the long-term goal in computer vision: an intelligent agent that can comprehend human instructions in natural language and execute them in visual environment. Once achieved, it will benefit various human-computer interaction applications such as visual Q&A [2], visual dialog [3], and robotic navigation [4]. To achieve this, a necessary step is to extend the current object detection system from fixed-sized inventory of words to open-vocabulary sentences, that is, grounding natural language in images [5].

Thanks to the advance of visual deep features [6] and neural language models [7], recent studies show promising results on scaling up visual grounding to open-vocabulary scenario, such as thousands of object categories [1], [8], relationships [9], and phrases [10]. However, grounding natural language (cf. Fig. 1a) is still far from satisfactory as the key is not only to associate related semantics to the target visual object, but also to distinguish it from the contextual objects, especially those of the same category. For example, as shown in Fig. 1a, to ground the referring expression "a black dog on the left of the tree", we need to first detect objects in the image and then distinguish the referent "black dog" from other objects especially those with the same category "golden dog" using the context "black", "left of the tree".

To make a successful discrimination between context and referent, we need to parse the language into corresponding semantic components. As illustrated in Fig. 1b, current state-of-the-art models [11], [12], [13] learn to parse a sentence into the (subject, predicate, object) triplets, and the referent grounding score is the sum of the three grounding scores. The intuition behind these compositional methods is that the parsing helps to divide the original problem into easier sub-tasks, *i.e.*, finding the contextual regions that grounded by "predicate" and "object" semantics is apparently helpful to localize the referent. However, we argue that the above triplet composition for a sentence is still too coarse. For example, it is meaningful to parse short sentences such as "person riding bike" into triplets, as it has a clear grounding for individual "person", "bike", and their relationship; but it is problematic for general longer sentences with adjective clause, *e.g.*, it is still difficult to parse the following long sentence into one triplet: "a black dog on the left of the tree which is bigger than others".

In this paper, we propose a fine-grained natural language grounding model called **R**ecursi**v**e **G**rounding **Tree** (RvG-Tree). The key motivation is to decompose any language sentence into semantic constituents in a recursive way, that is, every node or the root of every sub-tree can be further decomposed and the decomposition stops at the leaves which are the single words. As illustrated in Fig. 1c, "black dog" can be decompose into "black" and "dog", and thus the compositional confidence for "black dog" can be accumulated by "something is black" and "something is a dog". Therefore, by using RvG-Tree, we can accumulate the grounding confidence score from the lower layers which are relatively simpler grounding sub-tasks. Compared to previous methods that rely on sentence embedding features, RvG-Tree offers an explainable way of understanding how

- *R. Hong is with the School of Computer & Information, Hefei University of Technology, Hefei, China. E-mail: see https://sites.google.com/site/homeofrichanghong/*
- *D. Liu is with the University of Science and Technology of China, Hefei, China. E-mail: liudq@mail.ustc.edu.cn*
- *X. Mo is with the School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore. E-mail: moxy@ntu.edu.sg*
- *X. He is with the School of Information Science and Technology, University of Science and Technology of China. E-mail: xiangnanhe@gmail.com*
- *H. Zhang is with the School of Computer Science & Engineering, Nanyang Technological University, Singapore. E-mail: see http://www.ntu.edu.sg/home/hanwangzhang/*

Manuscript received xx xx, xxxx; revised xx xx, xxxx.

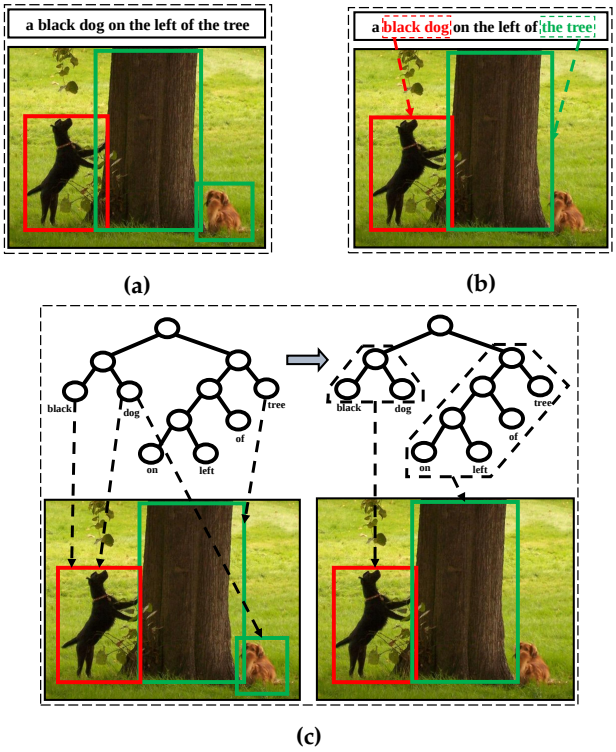


Fig. 1: (a) A typical task of grounding the natural language "a black dog on the left of the tree" in the image, represented as a set of objects represented as bounding boxes. The output should be the "dog" grounded with the red box. (b) Most recent advances focus on simple compositions of the language such as (subject, predicate, object) triplet. (c) Our RvG-Tree decomposes the sentence into more fine-grained compositions, and then accumulates the grounding confidence score in a bottom-up fashion.

the language is comprehended in visual grounding. One may argue that such binary decomposition into over fine-grained would be harmful, due to the fact that not all nodes refer to the target object. For example, "the tree". We tackle this case by learning a node classifier whether returns visual features for future grounding at higher nodes, or a grounding score that should be accumulated. Thanks to this design, our RvG-Tree is generic and flexible and thus can be applied in longer natural language sentences.

The technical overview of RvG-Tree is illustrated in Fig. 2. Inspired by the recent progress on tree structure construction for sentence representations [14], we propose to learn RvG-Tree in a bottom-up fashion, by dynamically merging any two adjacent nodes. Specifically, we start from leaf nodes which are words, where the two merged nodes are chosen based on their association score (*e.g.*, "black" and "dog"). Then, the merged and un-merged nodes are flushed to the next merging layer. Finally, the construction is complete when there are only one node left in the pool. Given an RvG-Tree constructed from a sentence, we design a recursive grounding score function that accumulates the grounding confidence from leaf to root. Considering any sub-tree with one root and two children nodes, we first use the node classifier to determine which children node is the *score* or feature node; the score node returns the grounding score from its own sub-tree, and the feature node returns the estimated context visual regions, which are soft-attention regions that the attention weights are normalized scores returned from its own sub-tree. The overall grounding score contains two non-differentiable decision-making processes: 1) the node merging process — choosing the highest association score in the pool — in the RvG-Tree construction, and 2) the score and feature node classification in the recursive grounding score calculation. To this end, we use Gumbel-Softmax [15] with proper expert supervision to make the overall architecture fully-differentiable, *i.e.*, standard SGD can be applied in the discrete decisions.

We perform extensive experiments on three challenging referring expression grounding datasets: RefCOCO [16], RefCOCO+ [16], and RefCOCOg [5]. Compared to existing grounding models, RvG-Tree is the first model that has totally transparent visual reasoning process for grounding and achieves comparative or even better performances.

Our contributions are summarized as follows:

- We propose RvG-Tree: an explainable vision-language reasoning model for visual grounding.
- RvG-Tree introduces a novel tree structure to parse the language input and calculates the grounding score in an efficiently recursive fashion, allowing machines to understand natural language in a way similar to the language constitution.
- RvG-Tree is designed to be fully-differentiable and thus it can be trained efficiently with standard SGD.

2 Related Work

2.1 Grounding Natural Language

Referring expressions are natural language statements describing the referent objects within a particular scene, *e.g.*, "the man on the left of the golden dog" or "the dog on the sofa". Grounding referring expression, which aims to localize the referring expression in a image, is also known as referring expression comprehension, and its inverse task is called referring expression generation [5]. Based on valid phrase grounding methods, referring expression grounding steps further to recognize the referent from other objects mentioned in the language input.

The task of grounding referring expression is to localize the region in the image given a referring expression. To solve this problem, joint embedding model is widely used in recent works [17], [18], [19]. They model the conditional probability $P(o|r)$, where r is the referent and o is the appropriate visual object. Instead of modeling $P(o|r)$ directly, others [5], [16], [20], [21], [22], [23], [24] compute $P(r|o)$ by using the CNN-LSTM structure for language generation. The visual region o maximizing $P(r|o)$ is considered to be the target region. Taking advantages of both the above mentioned approaches, Yu *et al.* [25] consider the joint-embedding model as a listener, CNN-LSTM as a speaker, and combine them to form a joint speaker-listener-reinforcer model to achieve state-of-the-art results. Instead of using holistic language feature to do referring expression grounding, some recent work decompose the language input into different parts. Modular Attention Network (MAttNet) [13] decomposes expressions into three modules related to subject appearance, location, and relationship to other objects, rather than treating them as a single unit. This model then

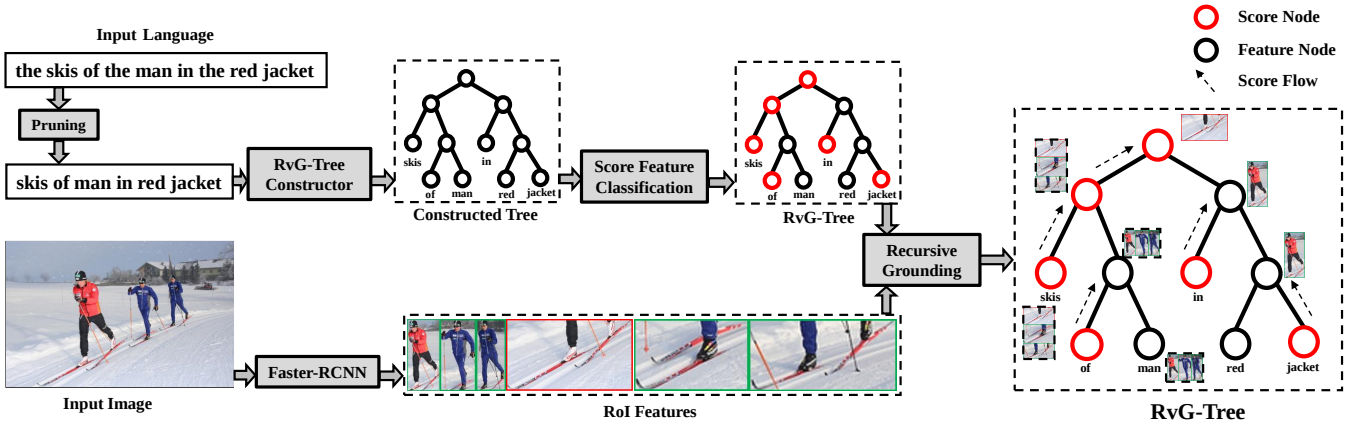


Fig. 2: The overview of using RvG-Tree for natural language grounding. Given an natural language sentence input, we first use NLP tools to prune the sentence (Section 3.2) and then construct RvG-Tree. Then, we have a score-feature classifier (Section 3.3) to determine each node to be the "score node" or "feature node", where the score node returns the recursive score and the feature node returns the feature (Section 3.3). The final score of the root node is accumulated recursively in a bottom-up fashion (Section 3.3) and the visual region with the highest score is considered as the grounding result. Note that all the nodes can be visualized by the corresponding region confidence scores and only qualitative regions are visualized.

calculates an overall score dynamically from all the three modules with weights learned from the language based attention. Visual attention has been used to facilitate the subject and relationship modules to focus on relevant image regions. Compositional Modular Network (CMN) [11] is a modular deep architecture which divides the input language into vector representations of subject, relationship, and object with attention and then integrates the scores of these three modules into the final score indicating which region is more qualified for the given language input. Separating an entire sentence into several components and analyzing these components using specific models makes the analysis more fine-grained.

However, it is worth noting that natural language has a latent hierarchical structure. Facilitating such latent structure information would make the grounding model more reasonable and explainable. Our model steps further in this direction by taking the latent hierarchical structure of the language into account. We automatically compose a tree structure to parse the language and then perform visual reasoning along the tree in a bottom-up fashion by accumulating grounding confidence scores.

2.2 Learning Tree Structures for Language

In their NLP community, learning tree structures for sentences is becoming more and more popular in recent years. Bowman *et al.* [26] built trees and compose semantics via a generic shift-reduce parser, whose training relies on ground-truth parsing trees. TreeRNNs combined with latent tree learning has been deemed as an effective approach for sentence embedding as it jointly optimizes the sentence embedding and a task-specific objective. For instance, Yogatama *et al.* [27] used REINFORCE algorithms [28] to train the shift-reduce parser without ground truth. Instead of the shift-reduce parsers, Maillard *et al.* [29] used a chart parser, which is fully differentiable by introducing a softmax annealing but suffers from $\mathcal{O}(n^3)$ time- and space-complexity. Gumbel Tree-LSTM is a parsing strategy proposed by [14], which introduces Tree-LSTM and calculates the merging score for each adjacent node pair based on a learnable query vector and greedily merges the best pair with the highest score in the next layer. They introduced Straight-Through Gumbel-Softmax estimator [15] to soften a hard categorical one-hot distribution into a soft distribution so as to enable end-to-end training. Comparison between above mentioned models on several datasets, which is done by [30], shows that Gumbel Tree-LSTM achieves the best performance. Our model facilitates the approach to learn the latent tree structure out from a flat language input to do visual reasoning for the natural language grounding task.

Tree structures for language have also been studied in the field of vision-language tasks. Xiao *et al.* [31] introduced the dependency parsing tree as a structural loss in visual grounding, thus the grounding results are expected to be more faithful to the sentence. Our work is fundamentally different from theirs as we explicitly perform grounding score calculation along the tree. In addition, note that our tree is more similar to constitution tree but not dependency tree. To minimize the biases in existing VQA datasets, Johnson *et al.* [32] proposed a diagnostic dataset CLEVR that tests a range of visual reasoning abilities. Questions in the dataset CLEVR are built using several categorical functions (*e.g.*, Filter, Equal and Relate) by composing these simple building blocks. Johnson *et al.* [33] proposed a method which contains two main modules: program generator and execution engine. The program generator takes a sequence of words as inputs and outputs a program as a sequence of functions. The resulting sequence of the functions is then converted to a syntax tree for the execution of visual reasoning by making use of the fact that the arguments of each function are known. Hu *et al.* [34] proposed an End-to-End Module Networks (N2NMNs) containing two components: a layout policy, which inputs deep representation of a question and outputs both a sequence of structural actions and a sequence of attentive actions, and a network builder, which takes these two sequences as input and outputs an appropriately

structured network to complete visual reasoning. All the above mentioned methods for VQA task seek to explore the latent structure of the input question.

3 RvG-Tree Model

We first define the problem of natural language grounding formally, and then introduce the RvG-Tree grounding model as illustrated in Fig. 2 as a walk-through example. Finally, we show how to train RvG-Tree as a neural network.

3.1 Problem Definition

We represent an image as a set of Region of Interest (ROI) features $\mathcal{I} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional feature vector, *e.g.*, extracted from any deep vision model such as Faster R-CNN [35]. Each ROI is a visual object detected in the image. We represent a natural language sentence as an m-length sequence $\mathcal{L} = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_m\}$, where $\mathbf{w}_i \in \mathbb{R}^b$ is a b-dimensional word embedding vector, *e.g.*, initialized from any word-vector models such as GloVe [36]. The task of grounding language $\mathcal{L}$ in image $\mathcal{I}$ can be represented as the following ranking problem:

$$\mathbf{x}^* = \underset{i}{\operatorname{argmax}}\, S(\mathbf{x}_i, \mathcal{L}), \tag{1}$$

where $S(\cdot)$ is a grounding score function that evaluates the association between region $\mathbf{x}_i \in \mathcal{I}$ and language $\mathcal{L}$.

Designing a good score function for Eq. (1) is not trivial because it is challenging to exploit the compositional nature of the language: parsing the sentence into semantic structures that capture the implied referent (*i.e.*, the target region) and the context (*i.e.*, regions that help to distinguish the referent out of others). Therefore, previous grounding models that only uses holistic sentence-level [5] or phrase-level [10] language features are straightforward but suboptimal. Recently, the triplet composition [11] is proposed to decompose the grounding score in Eq (1) into three subscores: referent (or subject), context (or object), and their pairwise relationship scores:

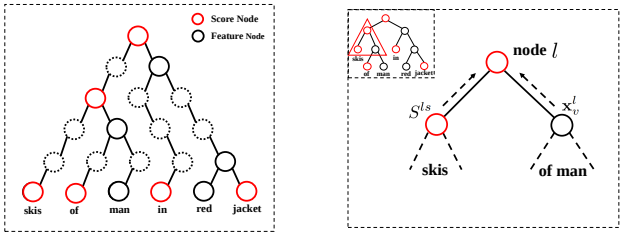
$$S(\mathbf{x}_i, \mathcal{L}) := S_s(\mathbf{x}_i, \mathbf{y}_s) + S_v(\mathbf{x}_v, \mathbf{y}_v) + S_p([\mathbf{x}_i, \mathbf{x}_v], \mathbf{y}_p), \tag{2}$$

where $\mathbf{y}_s$, $\mathbf{y}_v$, and $\mathbf{y}_p$ are the b-dimensional language features (the same dimension as the word embedding) for the 3 linguistic roles: referent, context, and relationship, respectively. They are computed by soft-attention weighted sum over the word vectors in the sentence, where the attention weights are word-relevance to each of the linguistic roles. $\mathbf{x}_v \in \mathbb{R}^d$ is the ROI feature for the context. As illustrated in Fig. 1a, take "a black dog on the left of the tree" as an example with perfect language parsing and visual detection, $\mathbf{y}_s = 0.5\mathbf{w}_{\text{black}} + 0.5\mathbf{w}_{\text{dog}}$, $\mathbf{y}_v = \mathbf{w}_{\text{tree}}$, and $\mathbf{y}_p = \mathbf{w}_{\text{left}}$, and $\mathbf{x}_v$ should be the ROI of "tree". Therefore, any region $\mathbf{x}_i$ of "dog" is expected to receive a higher score compared to the regions of other objects.

However, it is still not easy to obtain accurate $\mathbf{y}_s$, $\mathbf{y}_v$, $\mathbf{y}_p$, and $\mathbf{x}_v$ in Eq. (2), especially, when the language consists of more complex compositions such as "a black and white cat on top of the tree which is in front of a truck". The reasons are due to the error-prone modules as follows.

Language Composition: The *referent*, *context*, and *relationship* compositions produced by off-the-shelf syntactic parsers do not always correspond to intuitive reasoning of visual grounding. For example, the object of "a black and white cat on top of the tree which is in front of a truck" will be parsed, if perfectly, as "the tree which is in front of a truck", which is linguistically correct but visually difficult to learn the visual-semantic correspondence between a region and such a comprehensive expression. Therefore, we should further parse it into more fine-grained components for the ease of visual grounding.

Context Localization: Due to the prohibitively high cost of annotating both referent and context in images [12], we have to guess the context in a weakly-supervised way, that is, during training, the context object is not localized as the referent with ground-truth bounding boxes. Moreover, the context is not a single region but a multinomial combination of all the possible regions mentioned in the language. For example, how to compose a comprehensive representation $\mathbf{x}_c$ accounting for "black and white" and "on top of the tree which is in front of a truck" is challenging.



(a) Tree Construction (b) Recursive Unit

Fig. 3: (a). Given a flat sentence, RvG-Tree computes a score for every parent candidate indicating how qualified each candidate is to be merged in the next layer. By operating selection and merging recursively until RvG-Tree reaches the root node, the tree is constructed and will be used in the inference part. (b). RvG-Tree calculates grounding confidence scores in a recursive way. Taking the l-th node as an example, the grounding score returned by it is a combination of the scores of itself and its children.

3.2 RvG-Tree Construction

To address the two challenges introduced above, we propose to further decompose our grounding score in a recursive way by using a binary tree, allowing much more fine-grained visual reasoning. The motivations are two-fold: 1) the natural language can be generally divided into recursive components: we can always use attributive clause to modify a noun when necessary, and each clause can be recursively parsed into two linguistic components, such as the (subject, object), (attribute, subject), or (preposition, subject) pairs. 2) by using trees, we can do more fine-grained localization with simpler expressions and such simple grounding scores can be accumulated along the tree in a bottom-up fashion.

Before we construct the tree, we prune the sentence by discarding some determiners and symbols such as "a, an, another, any, both, each, either, those, that". We find that this pruning does not affect the overall performance while boost the speed. Similar to the method in [14], RvG-Tree calculates a score indicating how valid a composition is for every parent candidate. Composition here means to merge two adjacent nodes into a parent one. Based on the validity score,

the model recursively selects compositions in a bottom-up fashion, until it reaches the root representation. Fig. 3a is a walk-through example of RvG-Tree construction for the sentence "skis of man in red jacket" in Fig. 2. The first merging happens at "red" and "jacket", then the merged node together with other nodes: "skis", "of", "man", and "in", are the input for the next merging process, and "of" and "man" are merged. We repeat this process until we have two nodes left: one merged from "skis of man" and the other one merged from "in red jacket".

Formally, we first need to embed each node into features and then use them to decide which two nodes to merge in a computational way. We start from the leaf nodes, where each one is represented as the word embedding $\mathbf{y}_l$ in the sentence $\mathcal{L}$. To encode the contextual information of the words in sentence, we use a bi-directional LSTM (BiLSTM) [37] to obtain the initial l-th node feature $\mathbf{v}_l^{t=1}$ as:

$$\mathbf{v}_l^1 = \begin{bmatrix}\mathbf{h}_l^1 \\ \mathbf{c}_l^1\end{bmatrix} = \text{BiLSTM}(\mathbf{y}_l, \mathbf{h}_{l-1}^1, \mathbf{h}_{l+1}^1), \tag{3}$$

where $\mathbf{h}_l^1$ and $\mathbf{c}_l^1$ are hidden and memory cell vectors of the BiLSTM. Then, we can use $\mathbf{v}_l^1$ to merge two of them for the next layer $t = 2$ by using Eq. (5), which will be discussed soon. Next, we introduce how to obtain the node features for layers $t \leq 2$.

Without loss of generality, as shown in Fig. 3a, suppose "red" and "jacket" are merged as new node for the next layer $t = 2$, then all the other leaf nodes are upgraded to $t = 2$ for the next merging step:

$$\mathbf{v}_j^{t+1} = \begin{cases} \mathbf{v}_j^t & j < l \\ \left[\mathbf{h}^{t+1}; \mathbf{c}^{t+1}\right] = \text{TreeLSTM}(\mathbf{v}_j^t, \mathbf{v}_{j+1}^t) & j = l \\ \mathbf{v}_{j+1}^t & j > l \end{cases} \tag{4}$$

where $\mathbf{h}^t$ and $\mathbf{c}^t$ are the hidden and memory cell vectors from the Tree LSTM network (TreeLSTM) [38], which is a simple extension of the original LSTM by concatenating the children hidden states as the input hidden states.

Now, we introduce how to merge two adjacent nodes. We introduce a trainable parameter θ_p to measure the validity of a parent. Specifically, we use $\theta_p^T \mathbf{v}_\mathbf{l}^\mathbf{t}$ as the unnormalized validity score of a candidate parent representation in Eq. (4). Then, we decide whether to merge its two candidate children by selecting the largest (*i.e.*, argmax) softmax normalized score:

$$s_l = \frac{\exp(\theta_p^T \mathbf{v}_l^t)}{\sum_j \exp(\theta_p^T \mathbf{v}_j^t)}. \tag{5}$$

We repeat this procedure until we reach the root node of the tree, *i.e.*, the final RvG-Tree structure.

Note that the node feature embedding functions described in Eq. (3) and Eq. (4) are differentiable, but the merge procedure by using Eq. (5) is not, due to the greedy argmax. To tackle the discrete nature of the tree structure construction, we deploy the Gumbel-Softmax trick detailed in Section 3.4.1.

3.3 Recursive Grounding

Given the constructed RvG-Tree described in the previous section, we can accumulate the grounding confidence scores according to the language composition along the tree in a bottom-up fashion. Without loss of generality, suppose we are interested in calculating the l-th node (Fig. 3b), which has two children nodes: *score* node ls and *feature* node lv (which will be discussed soon). Then, the grounding score returned by the l-th node is defined in a recursive fashion as:

$$S^l(\mathbf{x}_i, \mathcal{L}_l) := \overbrace{S_s^l(\mathbf{x}_i, \mathbf{y}_s^l) + S_p^l([\mathbf{x}_i, \underbrace{\mathbf{x}_{lv}}_{\text{feature returned from node } lv}], \mathbf{y}_p^l)}^{\text{score calculated at node } l} + \underbrace{S^{ls}(\mathbf{x}_i, \mathcal{L}_{ls})}_{\text{score returned from node } ls}. \tag{6}$$

From the "divide & conquer" perspective, the "dirty" job (*i.e.*, conquer) is done by the "score calculated at node l" terms, and the "easy" ones (*i.e.*, divide) are just to ask the lv and ls children to give us "feature returned from node lv" and "score returned from node ls", and thus the reasoning can be performed in a bottom-up fashion. Interestingly, Eq. (6) can be viewed as a more generic and hierarchical formulation of the widely-used triplet composition [11], [13], [16] as in Eq. (2); however, the key difference is that our composition is achieved in an explicitly recursive way along the sentence, while the previous one is learned in an implicitly flat fashion.

Complexity. Compared to holistic or simple compositional methods such as Hu *et al.* [11] and Yu *et al.* [13], the proposed recursive grounding in Eq. (6) is more computationally expensive but the overhead is linear to sentence length and thus affordable. Suppose their computational cost is unit 1 and the sentence length is N, the number of score calculation is the number nodes: $\mathcal{O}(2N)$.

Next, we will discuss the design and notation details of Eq. (6) as follows.

3.3.1 Score Node & Feature Node Definition

According to the primal score of Eq. (1), a grounding score is to measure the association between any region $\mathbf{x}_i$ and a language sentence $\mathcal{L}$. However, as our recursive grounding will go through every word or sub-sequence in the sentence, it is not always reasonable to explicitly calculate the association. To this end, we introduce the score and feature nodes that are specially designed for recursive grounding, as illustrated in Fig. 3b.

Score Node. If a node l is a score node, its score will be accumulated to the higher layer. In particular, every root is a score node. Compared to the following introduced feature node, a score node calculates the grounding score according to Eq. (6) and pass it to the higher-layer node:

$$S^{ls} \leftarrow S^l, \tag{7}$$

where the score assignment denotes that the output of the score node is the score S^{ls} in Eq. (6). Thanks to the score nodes, we can relax the unreasonable cases in visual grounding that every language component should correspond to a visual region. We will further discuss this intuition later in Section 3.3.2.

Feature Node. If a node l is a feature node, it first adopts the same procedure as score node: calculate the grounding score according to Eq. (6), and then it further aggregates a

(a) Feature Node (b) Score Node

Fig. 4: Word cloud visualizations of what nodes are likely to be classified as (a) feature node or (b) score node. Every word frequency is increased by 1 if it is a leaf under the feature or score node. We can see that feature nodes are generally related to visible concepts while score nodes are generally not visible.

weighted sum over the region features, where the weights are normalized by the grounding score:

$$\mathbf{x} = \sum_i \frac{\exp\left(S^l(\mathbf{x}_i, \mathcal{L}_l)\right)}{\sum_j \exp\left(S^l(\mathbf{x}_j, \mathcal{L}_l)\right)} \mathbf{x}_i, \tag{8}$$
$$\mathbf{x}_{lv} \leftarrow \mathbf{x}.$$

Note that the above feature assignment indicates that the weighted feature $\mathbf{x}$ at the current node l is passed to its parent node and used as the output of the feature node $\mathbf{x}_{lv}$ in Eq. (6). In the point of view of the hierarchical feature representations any deep network, we should feed-forward the visual features in a bottom-up fashion, *i.e.*, the feature node. On the other hand, the score node is analogous to the loss that accumulated from intermediate features [39].

3.3.2 Score Node & Feature Node Classification

To determine whether child node 1 (or 2) is the feature node lv and the other one is the score node ls, we use the following binary softmax to be the probability of the feature node p_{v1}:

$$p_{v1} = \frac{\exp(\theta_v^T \mathbf{v}_1)}{\exp(\theta_v^T \mathbf{v}_1) + \exp(\theta_v^T \mathbf{v}_2)}, \tag{9}$$

where θ_v is the trainable parameter, $\mathbf{v}_{1/2}$ is the children node feature exactly as the same as in Eq. (4). Note that we also have the following probabilities:

$$p_{v2} = p_{s1} = 1 - p_{v1}, \quad p_{s2} = p_{v1}, \tag{10}$$

where $p_{s\cdot}$ is probability of score node. Similar to the discrete policy that causes non-differentiability in tree construction as in Eq. (5), we deploy Gumbel-Softmax to resolve this issue raised by Eq. (9).

Fig. 4 illustrates what nodes are likely to be classified as the feature or score nodes. We can see that nodes which have more visible leaf words like adjectives such as colors are more likely to be the feature nodes, and those which have more non-visible leaf words like relationships ("behind" and "sitting") are more likely to be the score nodes.

3.3.3 Language Feature

We use language feature to localize the corresponding visual regions referred in the language. Essentially, we would like to calculate the multimodal association between the visual features and the language features. We denote $\mathbf{y}_s^l$ as the language feature used to associate with a single region feature (*i.e.*,$\mathbf{x}_i$) and $\mathbf{y}_p^l$ to associate with a pairwise visual feature (*i.e.*, $[\mathbf{x}_i, \mathbf{x}_{lv}]$). Specifically, the language feature is calculated as a soft-weighted sum of the corresponding word embeddings in the sub-sequence $\mathcal{L}_l$:

$$\mathbf{y}_s^l = \sum_{i=1}^{|\mathcal{L}_l|} \alpha_{si} \mathbf{w}_i, \quad \mathbf{y}_p^l = \sum_{i=1}^{|\mathcal{L}_l|} \alpha_{pi} \mathbf{w}_i, \tag{11}$$

where $\mathbf{w}_i$ is the i-th word embedding vector and α is the word-level attention weights:

$$\alpha_{si} = \frac{\exp\left(\theta_s^T \mathbf{v}_i\right)}{\sum_j \exp\left(\theta_s^T \mathbf{v}_j\right)}, \quad \alpha_{pi} = \frac{\exp\left(\theta_p^T \mathbf{v}_i\right)}{\sum_j \exp\left(\theta_p^T \mathbf{v}_j\right)}, \tag{12}$$

where θ is the trainable parameter and $\mathbf{v}$ is the leaf node feature that is the same as in Eq. (4)&(9). The reason why we need word-level attention for extracting the language feature is because not all the words are related to the visual feature. Hence, suppressing irrelevant words will help the multimodal association. It is worth noting that the reason why we use the sum of word-level embeddings while not the BiTreeLSTM hidden vectors is because the latter is too diverse in the limited sentence pattern in our training scenario; however, the former is more stable as the diversity of words is significantly smaller than that of word compositions of sentences.

3.3.4 Score Functions

The score function S_s^l indicates how likely $\mathbf{x}_i$ is the referent given $\mathbf{y}_s^l$ and S_p^l shows how the pair-wise visual feature $[\mathbf{x}_i, \mathbf{x}_{lv}]$ matches the relationship described in $\mathbf{y}_p^l$. Formally, they are defined as the following simple two-layer MLPs [11], [12]:

$$S_s^l(\mathbf{x}_i, \mathbf{y}_s^l) = \theta_{s1}^T \left[\text{L2Norm}\left(\theta_{s2}^T \mathbf{x}_i \odot \mathbf{y}_s^l\right)\right], \tag{13}$$

$$S_p^l([\mathbf{x}_i, \mathbf{x}_{lv}], \mathbf{y}_p^l) = \theta_{p1}^T \left[\text{L2Norm}\left(\theta_{p2}^T [\mathbf{x}_i, \mathbf{x}_{lv}] \odot \mathbf{y}_p^l\right)\right] \tag{14}$$

where the θs are the trainable parameters, $\odot$ is element-wise multiplication, L2Norm is used to normalize the vector as unit L2 norm.

3.3.5 Leaf Case

The recursive Eq. (6) will arrive at the one layer above the leaves, that is, when the two children nodes are words, we may encounter extreme cases that the words cannot be visually grounded such as "with", "of", and "is", causing difficulties in interpreting the scores S_s^l and S_p^l. Fortunately, in these cases, the score functions defined in Eq. (13) and (14) will calculate similar scores for each region, as none of them is visually related to the words. As a result, accumulating such trivial scores would not affect the overall ranking score.

When the recursion in Eq. (6) arrives at the leaves, S^{ls} will calculate a grounding score for an empty sentence. Thus, we define the exit of the recursion as:

$$S^{ls}(\mathbf{x}_i, \mathcal{L}_{ls}) = S^{ls}(\mathbf{x}_i, \phi) = 0, \quad \text{if } l = \text{ leaf}. \tag{15}$$

Algorithm 1: RVG-TREE GROUNDING PIPELINE

Input : Language sentence, Image features
Prune the language sentence;
Embed words into node features as Eq. (3);
Initialize grounding scores as Eq. (15);
for Each layer **do**
  Find which two adjacent nodes to merge as Eq. (5);
  Classify score node and feature node as Eq.(9);
  Update node features as Eq. (4);
  Update grounding scores for each node ad Eq. (6);
end
Output: Grounding score of root

3.4 RVG-TREE Training

The inference of the RVG-TREE model is summarized in Algorithm 1. The model can be trained end-to-end in a supervised fashion when the ground truth region $\mathbf{x}_g$ referred in the language $\mathcal{L}$ is given. To distinguish the referring region from other regions, the model is expected to output a high $S(\mathbf{x}_g, \mathcal{L})$ for the ground-truth region and a low $S(\mathbf{x}_i, \mathcal{L})$ whenever $i \neq g$. Therefore, we train our model using the following cross-entropy loss:

$$L(\Theta) = -\log \frac{\exp\left(S(\mathbf{x}_g, \mathcal{L})\right)}{\sum_i \exp(S\left(\mathbf{x}_i, \mathcal{L}\right))}, \tag{16}$$

where Θ denotes all the trainable parameters in our model. Note that this loss is also called the Maximum Mutual Information (MMI) training in the pioneering work [5], as it is the same as maximizing the mutual information between the referent region and others (with the assumption of a uniform prior). The purpose is to ground the referent unambiguously by penalizing the model if it grounds other regions with high scores. Within a similar spirit, we can reformulate Eq. (16) into a large-margin triplet loss as in [25] with hard-negative sample mining. However, in our experiments, we observed only marginal performance gain but more tricky learning rate adjustment. Thus, we use Eq. (16) as the overall training objective in this paper.

3.4.1 Straight-Through Gumbel-Softmax Estimator

Note that it is prohibitive to use stochastic gradient descent (SGD) that back-propagates gradients of Eq. (16) to update the parameters of our model. The reason is that the gradients are blocked in the steps that make discrete policies where we greedily choose a parent node according to Eq. (5) and decide which child is the feature node according to Eq. (9). To bridge the gradients over the gap raised by the discrete policies, we deploy the Straight-Through (ST) Gumbel-Softmax estimator [15], which takes different paths in the forward and backward propagation by replacing the discrete argmax function with a differentiable and re-parameterized softmax function. Formally, given unnormalized probabilities $\{p_1, \ldots, p_n\}$, a sample $b_i \in \{b_1, \ldots, b_n\}$ from the Gumbel-Softmax distribution is drawn by $b_i \sim \pi_i$:

$$\pi_i = \frac{\exp((\log(p_i) + g_i)/\tau)}{\sum_{j=1}^{n} \exp((\log(p_j) + g_j)/\tau)}, \tag{17}$$

where $g_i = -\log(-log(u_i))$ and $u_i \sim \mathrm{Uniform}(0, 1)$. g_i is called the Gumbel noise perturbing each $\log(p_i)$ and τ is a temperature parameter which diminishes to zero, a sample from the Gumbel-Softmax distribution becomes a *cold* one resembling the one-hot sample.

Then, the straight-through (ST) gradient estimator is used as follows: in the forward propagation, b_i is sampled by argmax; in the backward propagation, b_i has a continuous value:

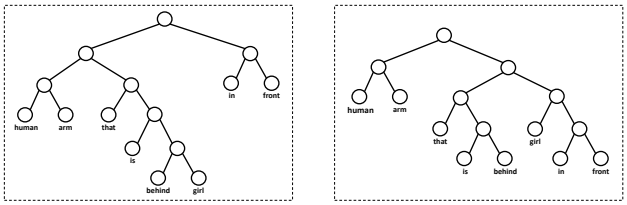
$$b_i = \begin{cases} \mathrm{argmax}_j\, b_j, & \text{forward prop} \\ \pi_i, & \text{backward prop} \end{cases}. \tag{18}$$

Intuitively, the estimator applies some random explorations (controlled by u_i) to select the best policy greedily in the forward pass, and it back-propagates the errors to all policies with a scaling factor π_i. Note that the noise in forward propagation of Eq. (18) is turned off in the test phase. Though this ST estimator is biased, it is shown to perform well in previous work [40] and our experiments. Gumbel-Softmax and its ST estimator is a re-parameterization trick for feature-based random variable where the output of a random choice is a feature while not a discrete layout. Thanks to this trick, the ST estimator can be considered as a soft-attention mechanism that efficiently back-propagates errors for all possible discrete tree structures, smartly avoiding from sampling the prohibitively large layout space such as REINFORCE [28]. In our experiments, we found that REINFORCE with Monte-Carlo sampling does not converge with even very small learning rate such as 1e-6.

3.4.2 Supervised Pre-Training

Minimizing the loss function in Eq. (16) from scratch is challenging: one need to simultaneously learn the all the parameters, especially those for the tree construction and feature/score node selection policies, which may suffer from a weaker stability and be trapped to a local optimum, greatly affecting the performance of our RVG-TREE. Therefore, we would like to apply a common practice: we first use the supervised pre-training to find a fair solution, *i.e.*, a good exploitation, and then use the end-to-end straight-through Gumbel Softmax as weakly supervised fine-tuning to achieve better exploration. However, there is no such an expert policy to generate a RVG-TREE like binary tree. To tackle this challenge, we borrow a third-part toolkit: Stanford CoreNLP (SCNLP) [41], which contains a constituency parser that takes a cleaned flat sentence as input and outputs a multi-branch constituency tree, where the children of a node are words constituting a phrase, *e.g.*, "furry black dog". SCNLP cleans the input sentence using *pos tag* before feeding it into the constituency parser by discarding punctuations and articles that bring unnecessary redundancy in the tree.

To transform the multi-branch constituency tree into a binary one, we apply a simple separation rule: for a subtree with n children, from left to right, we group every two consecutive words that constitutes a sub-binary-tree, and the left one, if any, is upgraded to be a single sub-tree with itself as the root. For example, the five children "a", "furry", "and", "black", "dog" are separated into "a furry", "and black", and "dog", and are further merged to "a furry and black" and "dog". Thus, we use this binary tree as the expert layout to train Eq. (5) with supervision. Fig. 5 illustrates



(a) Expert Tree **(b) Fine-tuned Tree**

Fig. 5: Examples of binary tree structures from (a) SC-NLP constituency parser and (b) overall fined-tuned by Straight-Through Gumbel-Softmax estimator. We can see that the tree structure is more meaningful after fine-tuning.

the differences between an expert tree and a resultant tree after fine-tuning. First, the expert rules divides the sentence into two chuncks that are difficult for grounding: "human arm that is behind girl" and "in front"; however, after fine-tuning, we can construct the tree in a more meaningful way: "human arm" that is the referent and "that is behind girl in front" as the context to be further parsed.

4 EXPERIMENTS

We conducted extensive experiments on three benchmarks of referring expression comprehension, *i.e.*, grounding the referent object described in the language. The motivation of our experimental design is to answer the following three questions:

- Is the tree structure better than holistic and triplet language models?
- Is the recursive grounding score effective?
- Is RVG-TREE explainable?

4.1 Datasets

RefCOCO [16]. It contains 142,210 expressions, which are collected using an interactive game, for 50,000 object instances in 19,994 images. All expression-referent pairs in this dataset are split into four mutually-exclusive parts: train, validation, Test A and Test B. This dataset allots 120,624 and 10,834 pairs to the train and validation part, respectively. Test A is alloted with 5,657 images, where each image has multiple people and Test B contains the rest 5,095 expression-referent pairs, where each image contains multiple objects.

RefCOCO+ [16]. It contains 141,564 referring expressions for 49,856 referents in 19,992 images. The referring expressions are collected in the same way as RefCOCO. It describes the referents with only appearance information by excluding absolute location words, making it different from RefCOCO. The train, validation, Test A, and Test B sections mutually exclusively contain 120,191, 10,758, 5726, and 4,889 expression-referent pairs, respectively.

RefCOCOg [5]. It contains 95,010 expressions for 49,822 referents in 25,799 images. The expressions containing both appearance and location expressions, which are collected in a non-interactive way, are longer than those in RefCOCO and RefCOCO+. There are two kinds of data separations for RefCOCOg. The first one [5] has no testing split released, so most recent works evaluate their models on the validation section. It is worth noting that the first partition randomly separates objects into training and validation sets, which leads to the fact that the images in both training and validation sets are not mutually exclusive. The second one [20] randomly splits images into training, validation and testing sets. We report our experimental results on both of them. RefCOCOg has significantly richer language and hence is more challenging than the previous two datasets.

4.2 Settings and Metrics

We set the length of each sentence in RefCOCO, RefCOCO+, and RefCOCOg, to 10, 10, 20, respectively, since such lengths can cover almost 95 percent sentences on all datasets. We use 'pad' symbol to pad expression sequences whose lengths are less than the set length. We use three specific vocabularies for the three datasets and the vocabulary sizes are 1,969, 2,596, and 3,314 for RefCOCO, RefCOCO+ and RefCOCOg, respectively. Word frequency in vocabularies was counted in all expressions and we discarded them that appeared less than 5 times in the entire dataset, then we replaced them with 'unk' in the vocabulary. Note that these 'unk's were still evaluated with the merge score in Eq. (5) and involved in the Gumbel-Softmax in training and softmax in test with zero-out mask, that is, the 'unk's were not considered in greedy merge. We used GloVe pre-trained word vectors [36] to initialize our word vectors. However, random initialized word vectors did not significantly degrade the performance.

We used ROI visual features annotated by MSCOCO for all three datasets. These ROIs are represented by 2048-d vectors, which are the fc7 output of a ResNet-101 based Faster-RCNN [35] trained on MSCOCO, 1024-d vectors, which are the pool5 output of the same Faster-RCNN, and 5-d vectors, which indicate the location of ROI in an image. Note that the goal of our experiments is to diagnose the visual reasoning capability of the grounding model, therefore, we did not use the most recent strong visual attribute features as in [13]. However, RVG-TREE is compatible to any visual feature input.

We used Adam [42] with initial learning rate 0.001, $\alpha = 0.8$, $\beta = 0.999$, and $\epsilon = \mathrm{e}^{-8}$, as our optimizer. We set 128 images to mini-batch size. For each sentence grounding, we calculated the intersection-over-union (IoU) of the selected bounding box with the ground-truth bounding box and considered the one with IoU larger than 0.5 as correct. We compute the fraction of correctly grounded test expressions as the grounding accuracy (*i.e.*, Top-1 Accuracy).

4.3 Ablative Studies

We conducted extensive ablative studies of RVG-TREE to justify our proposed design and training strategy. The ablations and their motivations are detailed as follows.

- **Chain**: We used BiLSTM to encode the sentence. Every word has two representations: 1) the 2048-d concatenation of its corresponding two-directional LSTM hidden vectors, and 2) the 300-d word embeddings. The first representation is used to calculate the word-level soft-attentions and then the language feature is represented as the soft-attention weighted average of the word embeddings. This ablation ignores the structure information of the language.

- **RvG-Tree-Fix**: We used TreeLSTM to encode the sentence. The binary tree is the constituent parsing tree result from Stanford Parser [43]. Similar to Chain, every word has 1) word embedding representations and 2) LSTM hidden state representations.
- **RvG-Tree-Scratch**: This is the full RvG-Tree model without the binary tree expert supervision, *i.e.*, the tree is constructed from scratch.
- **RvG-Tree/Node**: The tree is constructed in the same way as the full RvG-Tree model. The difference is that we ignore the first two scores in Eq. (6). We used this ablation to justify that the node-level score is an essential complementary to the bottom-up score accumulation.
- **RvG-Tree/S**: This is the RvG-Tree model without accumulating the score form the score node. That is, we ignore the S^{ls} in Eq. (6).
- **RvG-Tree/F**: This is the RvG-Tree model without the node view pairwise score S_p^l in Eq. (6). This ablation discards the visual feature returned by the feature node.

Table 1 shows the grounding accuracies of the ablative methods on the three benchmarks. We can have the following observations:

1) On all datasets, RvG-Tree-Fix outperforms Chain. This is because that the tree structure is more suitable for language decomposition, that is, the tree sentence feature captures more structure semantics than the holistic language feature, especially for longer sentences such as RefCOCOg. However, we should note that the improvement is limited. We believe that this is due to that the RvG-Tree-Fix is essentially still a holistic language representation as only the root embedding is used in visual reasoning. This motivates us to design grounding score function that exploits the tree structure explicitly.

2) If we delete the node view score, RvG-Tree/Node is significantly worse than the full RvG-Tree. The reasons are two-fold: first, the node view score is an independent score to correct, if any, wrong grounding confidence passed from children nodes; second, the node view offers a more comprehensive linguistic view compared to its children view, as its related sub-sequence is a joint set of the two children.

3) The reason why RvG-Tree/S is worse than RvG-Tree is because the grounding score is not accumulated. This demonstrates that compositional visual reasoning is crucial for the task of grounding natural language.

4) Without the pairwise score calculated from the visual feature returned by the feature node, RvG-Tree/F is inferior to RvG-Tree. This demonstrates that the pairwise relationship is essential for distinguishing the referent from its context. This is especially useful for longer sentences in RefCOCOg: RvG-Tree considerably higher than its non-pairwise counterpart RvG-Tree/F.

5) We can see that tree construction from scratch generally fails. This is not surprising as it is quite challenging to learn language composition without any prior knowledge. Note that this observation also agrees with many works in reinforcement learning that requires supervised training as the teacher forcing [34].

4.4 Comparison with State-of-the-Arts

We compared RvG-Tree with state-of-the-art referring expression grounding models published in recent years. In light of whether the model requires language composition, these comparing methods can be categorized as follows: 1) the resultant region is the one that can generate the referring expression sentence with maximum probability (by maximizing a posteriori probability), such as the pioneering **MMI** [5], **Attribute** [19], and the **Speaker** [25] and Listener [25]. Though the score mechanism exploits language composition during the sentence generation, it does not consider the different visual regions while generation. 2) localization based grounding methods that only use holistic language features, such as **NegBag** [20]. 3) localization based grounding methods that use language composition such as **CMN** [11], **VC** [12], **MAttN** [13]. Note that RvG-Tree belongs to the family of the last localization based grounding models, but its language composition is much more fine-grained than the previous state-of-the-arts.

4.4.1 Results on Ground-Truth Regions

From the results on RefCOCO, RefCOCO+, and RefCOCOg in Table 3, we can see that RvG-Tree achieves the state-of-the-art performance. We believe that the improvement is attributed to the recursive grounding score along the tree structure. First, on all datasets, RvG-Tree outperforms all the other sentence generation-comprehension methods: MMI, Attribute, Speaker, Listener, that do not consider language structure and visual context. Second, RvG-Tree outperforms the triplet compositional models such as CMN and VC. The improvement is attributed to the fact that RvG-Tree recursively applies the triplet-like grounding score along the tree structure, and hence the grounding confidence is more comprehensive than those methods. This demonstrates the effectiveness of the recursive fashion of the compositional grounding.

As shown in Fig. 6, we illustrate some qualitative correct grounding results, the learned tree structure, and their intermediate grounding process. Each intermediate node is visualized by the soft-attention map for the contextual feature if it is a feature node, or the score map of regions if it is a score node. We can see that most of the grounding results show reasonable intermediate results. For example, in the tree of "baseball player swinging bat at baseball", "baseball player" on the left sub-tree scores high value for both of the player regions; however, by the help of the right sub-tree "swinging bat at baseball" which scores higher for the person who is "swinging", RvG-Tree can pinpoint the correct referent in the end. This is intuitively similar to human reasoning and the purpose of using attributive clause in English. Moreover, the classification of score and feature node also sheds some light on the tree structure. For example, due to that the goal is to distinguish the baseball player, then the base player itself is working as a context feature vector, *i.e.*, feature node, and the score should be thus dominated by the score node with words "swinging bat at baseball". If the image contains more than one object class, the top score node is usually connected with the referent, *e.g.*, "backpack" and "bike". We also note that some of the tree structure is trivially deep, by always merging the last

	RefCOCO			RefCOCO+			RefCOCOg	
	val	testA	testB	val	testA	testB	val	test
Chain	80.14	80.54	79.61	65.77	66.80	60.97	72.77	71.69
RvG-Tree-Fix	81.52	80.77	80.53	66.33	68.16	62.53	73.69	73.07
RvG-Tree-Scratch	79.65	79.22	79.33	65.01	66.12	61.12	71.83	72.00
RvG-Tree/Node	82.93	82.41	82.30	67.76	69.33	64.47	74.10	74.36
RvG-Tree/S	82.24	81.12	80.91	67.32	68.87	64.05	74.21	73.18
RvG-Tree/F	82.50	81.79	81.49	67.48	69.37	64.29	74.12	73.98
RvG-Tree	83.48	82.52	82.90	68.86	70.21	65.49	76.82	75.20

TABLE 1: Top-1 Accuracy% of ablative models on the three datasets with ground-truth object bounding boxes.

	RefCOCO			RefCOCO+			RefCOCOg	
	val	testA	testB	val	testA	testB	val	test
Chain	72.01	75.55	67.34	59.44	62.78	53.90	64.05	64.73
RvG-Tree-Fix	73.59	77.23	68.81	60.80	64.30	54.29	64.87	65.04
RvG-Tree-Scratch	71.34	74.80	67.22	59.48	62.71	53.96	63.71	63.50
RvG-Tree/Node	74.66	77.23	68.50	62.28	66.30	55.72	65.58	65.53
RvG-Tree/S	74.22	76.89	68.02	61.95	65.77	55.01	65.12	64.99
RvG-Tree/F	74.13	77.28	68.21	62.38	65.98	55.34	65.55	65.25
RvG-Tree	75.06	78.61	69.85	63.51	67.45	56.66	66.95	66.51

TABLE 2: Top-1 Accuracy% of ablative models on the three datasets with detected object bounding boxes.

	RefCOCO			RefCOCO+			RefCOCOg		
	val	testA	testB	val	testA	testB	val*	val	test
MMI [5]	-	63.15	64.21	-	48.73	42.13	62.14	-	-
NegBag [20]	76.90	75.60	78.80	-	-	-	-	-	68.40
Attribute [19]	-	78.85	78.07	-	61.47	57.22	69.83	-	-
CMN [11]	-	75.94	79.57	-	59.29	59.34	69.30	-	-
VC [12]	-	78.98	82.39	-	62.56	62.90	73.98	-	-
Speaker [25]	79.56	78.95	80.22	62.26	64.60	59.62	72.63	71.65	71.92
Listener [25]	78.36	77.97	79.86	61.33	63.10	58.19	72.02	71.32	71.72
AccAttn [23]	81.27	81.17	80.01	65.56	68.76	60.63	73.18	-	-
MAttN* [13]	82.06	81.28	**83.20**	64.84	65.77	64.55	-	75.33	74.46
RvG-Tree	79.04	78.82	80.53	62.38	62.82	61.28	72.77	72.32	71.95
RvG-Tree*	**83.48**	**82.52**	82.90	**68.86**	**70.21**	**65.49**	**76.29**	**76.82**	**75.20**

TABLE 3: Top-1 Accuracy% of various grounding models on the three datasets with ground-truth object bounding boxes. * indicates that this model uses res101 features.

	RefCOCO			RefCOCO+			RefCOCOg		
	val	testA	testB	val	testA	testB	val*	val	test
MMI [5]	-	64.90	54.51	-	54.03	42.81	45.85	-	-
NegBag [20]	57.30	58.60	56.40	-	-	-	39.50	-	49.50
Attribute [19]	-	72.08	57.29	-	57.97	46.20	52.35	-	-
CMN [11]	-	71.03	65.77	-	54.32	47.76	57.47	-	-
VC [12]	-	73.33	67.44	-	58.4	53.18	62.30	-	-
Speaker [25]	69.48	72.95	63.43	55.71	60.43	48.74	59.51	60.21	59.63
Listener [25]	68.95	72.95	62.98	54.89	59.61	48.44	58.32	59.33	59.21
MAttN* [13]	72.96	76.61	68.20	58.91	63.06	55.19	-	64.66	63.88
RvG-Tree	71.59	76.05	68.03	57.56	61.07	53.18	63.45	63.73	63.38
RvG-Tree*	**75.06**	**78.61**	**69.85**	**63.51**	**67.45**	**56.66**	**66.20**	**66.95**	**66.51**

TABLE 4: Top-1 Accuracy% of various grounding models on the three datasets with detected object bounding boxes. * indicates that this model uses res101 features.

two nodes, for example, "man sitting on coach using laptop" and "backpack of last skier". So far, we cannot find out what the cause is, but it seems that such trivial structures are still reasonable for their corresponding grounding scenario.

Fig. 7 shows some failure cases of RvG-Tree performed on RefCOCOg. We can see that most of the errors are caused by the wrong comprehension of the nuanced visual relationships, especially those contain comparative semantics. For the example of "girl wearing pink shirt and jeans on bed next desk", from the intermediate results, we are happy to see that our model correctly grounds "pink shirt" and "jeans" but it fails to distinguish the which girl is "on" bed and "next" to the desk. In fact, both of the girls are "on" and "next" visually, that is, our visual model does not fail. However, the true semantic meaning should be "sit on", which is very challenging for current model to discover. This failure may shed some lights on our future direction. As another example, though the model successfully identifies the "walking" out of "person walking behind bus shelter", its final score is lower (though close) than something that is "behind". This reveals some the drawback of RvG-Tree is that more linguistic prior knowledge should be used. For example, if we can identify the referent adjective is "walking", we may assign higher weights on the "walking person" but not the background "behind". Some failures are due to the imperfect visual recognition models for human actions, for example, "catching". This kind of failures is expected to be resolved by more fine-grained human action recognizer using parsed human bodies.

4.4.2 Results on Detected Regions

So far, our results are on grounding tasks with ground-truth object bounding boxes, that is, each visual region is guaran-

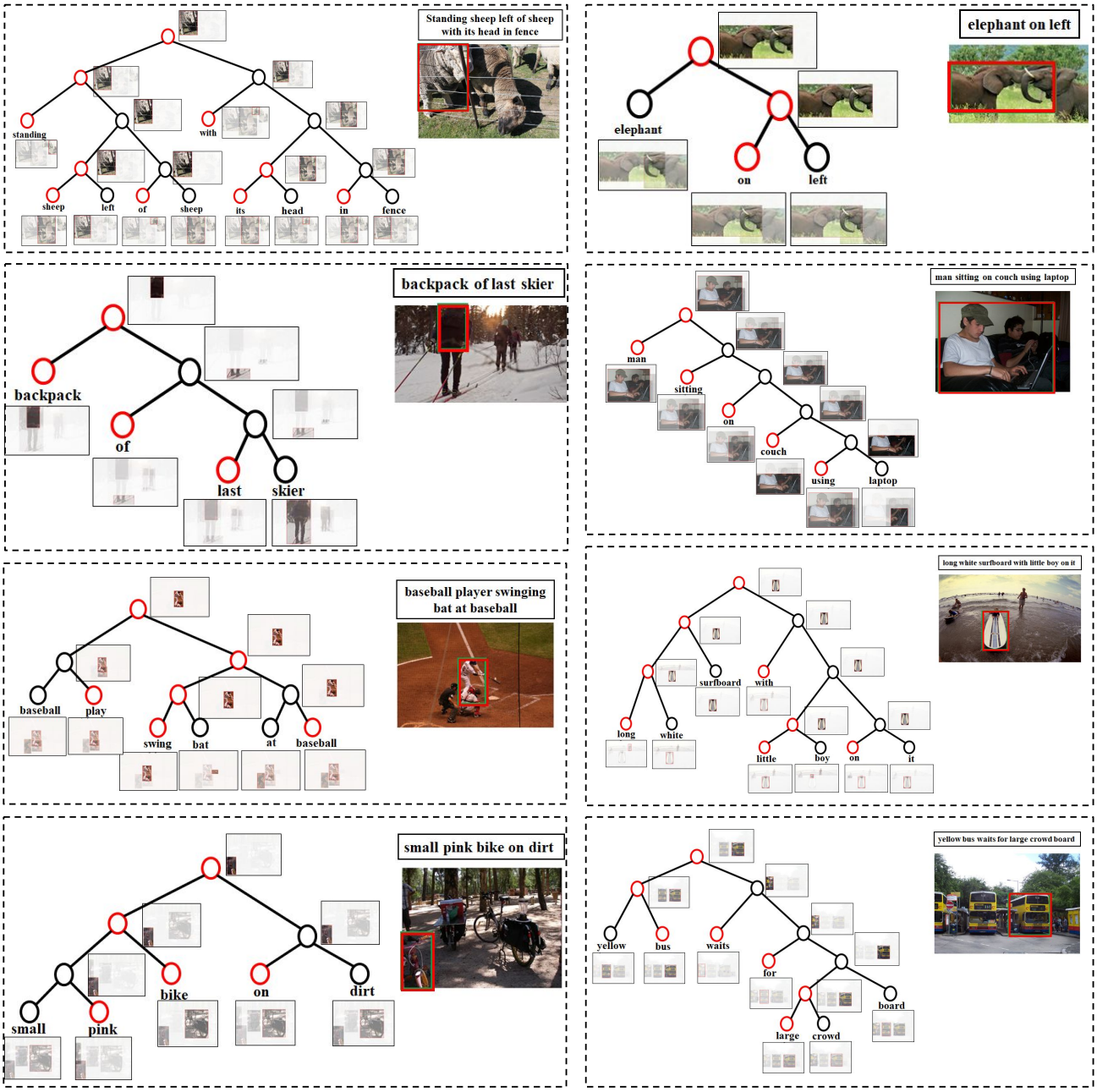


Fig. 6: Qualitative results of correct grounding results from RefCOCOg. Red circle is the score node and black circle is the feature node. Each node is visualized by the score map or attention map of different regions: larger transparency indicates lower score. The ground-truth region is in green box and the result is in red box.

teed to be a valid object. Though this setting eliminates the errors from imperfect object detection and hence allows the algorithmic design to focus on the discrimination between similar regions, it is not practical in real applications, where we always use object detectors to obtain the image regions. Therefore, it is also necessary to evaluate our methods on the various number of detected bounding boxes using Faster-RCNN. We followed the classic MS-COCO detection task NMS to obtain 10 to 100 objects per image.

From Table 4, we can see that the performance of all methods have been dropped due to the imperfect bounding box detections. However, the observations discovered in previous ground-truth box experiments still hold in detected boxes. It is worth noting that the performance of methods without compositional reasoning: MMI, NegBag, Attribute, Speaker, and Listener, dropped the most. It is due to the fact that these models only learn language and region associations that are easily overfitted to the distribution bias of the ground-truth bounding boxes; when our test moves to the detected boxes, the bias no longer holds. Compared to other compositional reasoning models: CMN, VC, and MAttN, our RvG-Tree is better. This demonstrates the robustness of our model to noisy visual regions.

5 Conclusions

In this paper, we propose a novel model called Recursive Grounding Tree (RvG-Tree) that localizes the target region by recursively accumulating the vision-language grounding

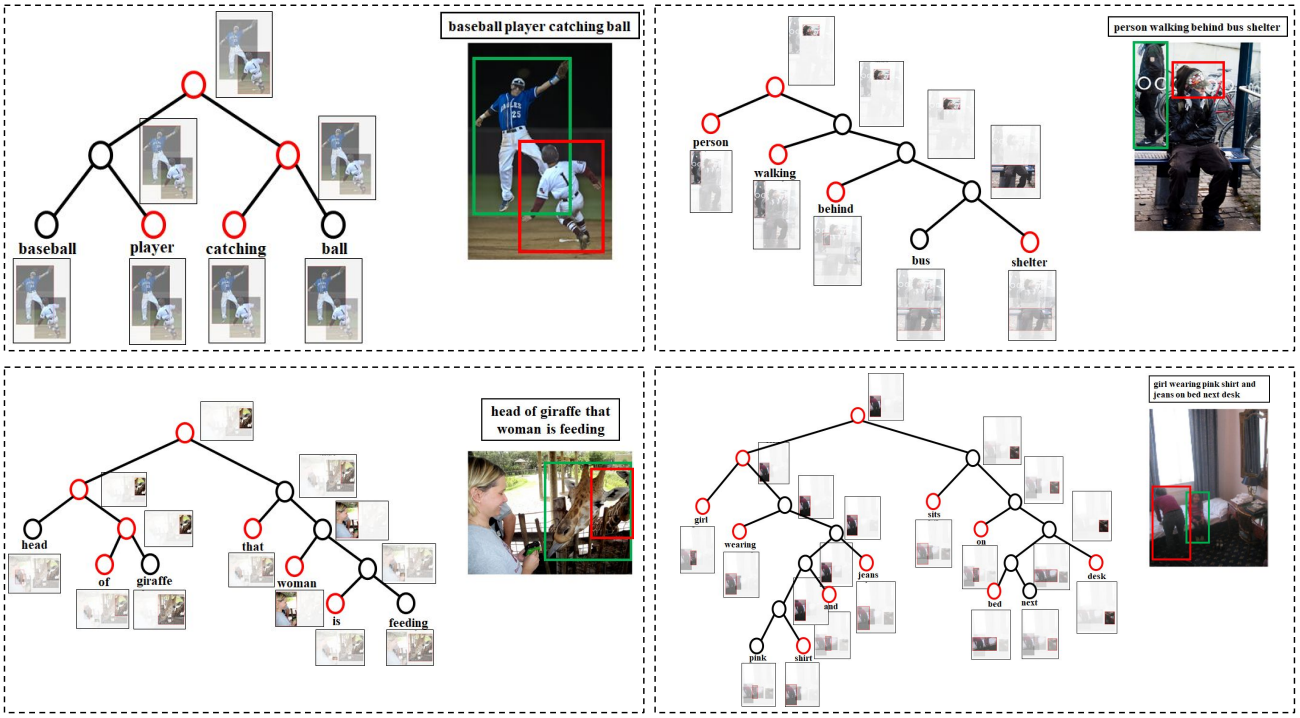


Fig. 7: Qualitative results of incorrect grounding results from RefCOCOg. Red circle is the score node and black circle is the feature node. Each node is visualized by the score map or attention map of different regions: larger transparency indicates lower score. The ground-truth region is in green box and the result is in red box.

scores. To the best of our knowledge, this is the first visual grounding model that leverages the entire language compositions. RvG-Tree learns to compose a binary tree structure, with proper supervised pre-training, and the final grounding score of the root is defined in a recursive way that accumulates grounding confidence from two children sub-trees. This process is fully differentiable. We conducted extensive ablative, quantitative, and qualitative experiments on three benchmark datasets of referring expression grounding. Results demonstrate the effectiveness of RvG-Tree in visual reasoning: 1) the complex language composition is decomposed into easier sub-grounding tasks, and 2) the overall grounding score can be easily explained by inspecting the intermediate grounding results along the tree. Therefore, compared to existing models, RvG-Tree is more compositional and explainable.

The key limitation of RvG-Tree is that the linguistic prior knowledge is not fully exploited. However, acquiring such knowledge is essentially important for machine comprehension of natural language and then grounding it based on the comprehension. Although we have demonstrated that using constituency parsers as prior knowledge can boost the performance, it is not sufficient. In future, we are going to add more linguistic cues into the recursive grounding score function to guide the confidence accumulation.

References

[1] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *CVPR*, 2017.
[2] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *ICCV*, 2015.
[3] A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra, "Learning cooperative visual dialog agents with deep reinforcement learning," in *ICCV*, 2017.
[4] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied question answering," in *CVPR*, 2018.
[5] J. Mao, J. Huang, A. Toshev, O. Camburu, A. L. Yuille, and K. Murphy, "Generation and comprehension of unambiguous object descriptions," in *CVPR*, 2016.
[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[7] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *ACISCA*, 2010.
[8] R. Hu, P. Dollár, K. He, T. Darrell, and R. Girshick, "Learning to segment every thing," in *CVPR*, 2018.
[9] H. Zhang, Z. Kyaw, S.-F. Chang, and T.-S. Chua, "Visual translation embedding network for visual relation detection," in *CVPR*, 2017.
[10] B. A. Plummer, A. Mallya, C. M. Cervantes, J. Hockenmaier, and S. Lazebnik, "Phrase localization and visual relationship detection with comprehensive image-language cues," in *Proc. ICCV*, 2017.
[11] R. Hu, M. Rohrbach, J. Andreas, T. Darrell, and K. Saenko, "Modeling relationships in referential expressions with compositional modular networks," *CVPR*, 2017.
[12] H. Zhang, Y. Niu, and S.-F. Chang, "Grounding referring expressions in images by variational context," in *CVPR*, 2018.
[13] L. Yu, Z. Lin, X. Shen, J. Yang, X. Lu, M. Bansal, and T. L. Berg, "Mattnet: Modular attention network for referring expression comprehension," in *CVPR*, 2018.
[14] J. Choi, K. M. Yoo, and S.-g. Lee, "Learning to compose task-specific tree structures," in *AAAI*, 2018.
[15] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *ICLR*, 2017.
[16] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg, "Modeling context in referring expressions," in *ECCV*, 2016.
[17] L. Wang, Y. Li, and S. Lazebnik, "Learning deep structure-preserving image-text embeddings," in *CVPR*, 2016.
[18] A. Rohrbach, M. Rohrbach, R. Hu, T. Darrell, and B. Schiele, "Grounding of textual phrases in images by reconstruction," in *ECCV*, 2016.
[19] J. Liu, L. Wang, M.-H. Yang *et al.*, "Referring expression generation and comprehension via attributes," in *CVPR*, 2017.

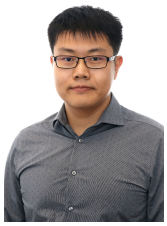
[20] V. K. Nagaraja, V. I. Morariu, and L. S. Davis, "Modeling context between objects for referring expression understanding," in *ECCV*, 2016.
[21] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell, "Natural language object retrieval," in *CVPR*, 2016.
[22] R. Luo and G. Shakhnarovich, "Comprehension-guided referring expressions," in *CVPR*, 2017.
[23] C. Deng, Q. Wu, Q. Wu, F. Hu, F. Lyu, and M. Tan, "Visual grounding via accumulated attention," in *CVPR*, 2018.
[24] Z. Yu, J. Yu, C. Xiang, Z. Zhao, Q. Tian, and D. Tao, "Rethinking diversified and discriminative proposal generation for visual grounding," in *IJCAI*, 2018.
[25] L. Yu, H. Tan, M. Bansal, and T. L. Berg, "A joint speakerlistener-reinforcer model for referring expressions," in *CVPR*, 2017.
[26] S. R. Bowman, J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, and C. Potts, "A fast unified model for parsing and sentence understanding," in *ACL*, 2016.
[27] D. Yogatama, P. Blunsom, C. Dyer, E. Grefenstette, and W. Ling, "Learning to compose words into sentences with reinforcement learning," *ICLR*, 2017.
[28] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, 1992.
[29] J. Maillard, S. Clark, and D. Yogatama, "Jointly learning sentence embeddings and syntax with unsupervised tree-lstms," *arXiv preprint arXiv:1705.09189*, 2017.
[30] A. Williams, A. Drozdov, and S. R. Bowman, "Learning to parse from a semantic objective: It works. is it syntax?" *arXiv preprint arXiv:1709.01121*, 2017.
[31] F. Xiao, L. Sigal, and Y. Jae Lee, "Weakly-supervised visual grounding of phrases with linguistic structures," in *CVPR*, 2017.
[32] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning," in *CVPR*, 2017.
[33] J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. L. Zitnick, and R. B. Girshick, "Inferring and executing programs for visual reasoning." in *ICCV*, 2017.
[34] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko, "Learning to reason: End-to-end module networks for visual question answering," in *ICCV*, 2017.
[35] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2016.
[36] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014.
[37] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *TSP*, 1997.
[38] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *EMNLP*, 2013.
[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.
[40] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," in *ICLR*, 2017.
[41] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *ACL*, 2014.
[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.
[43] M. Zhu, Y. Zhang, W. Chen, M. Zhang, and J. Zhu, "Fast and accurate shift-reduce constituent parsing," in *ACL*, 2013.

Richang Hong (M11) received the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2008. He is currently a Professor with the Hefei University of Technology, Hefei. His research interests include multimedia content analysis and social media, in which he has co-authored over 100 publications. He is a member of the ACM and an Executive Committee Member of the ACM SIGMM China Chapter. He was a recipient of the Best Paper Award at the ACM Multimedia 2010, the Best Paper Award at the ACM ICMR 2015, and the Honorable Mention of the IEEE TRANSACTIONS ON MULTIMEDIA Best Paper Award 2015. He was an Associate Editor of the IEEE Multimedia Magazine, Information Sciences and Signal Processing, Elsevier, and the Technical Program Chair of the MMM 2016, ICIMCS 2017, and PCM 2018.

Daqing Liu received the B.E. degree in Automation from Chang'an University, Xi'an, China, in 2016, and currently working toward the Ph.D. degree from the Department of Automation, University of Science and Technology of China, Hefei, China. His research interests mainly include computer vision and multimedia.

Xiaoyu Mo received the B.E. degree from Yangzhou University, Yangzhou, China, in 2015, the M.E. degree from Huazhong University of Science and Technology, Wuhan, China, in 2017. He was a research associate with Nanyang Technological University, Singapore, from Sep. 2017 to Jan. 2019. He is now a Ph.D. student in Nanyang Technological University. His research interests include consensus and autonomous vehicles.

Xiangnan He is currently a professor with the University of Science and Technology of China (USTC). He received his Ph.D. in Computer Science from National University of Singapore (NUS) in 2016, and did postdoctoral research in NUS until 2018. His research interests span information retrieval, data mining, and multimedia analytics. He has over 50 publications appeared in several top conferences such as SIGIR, WWW, and MM, and journals including TKDE, TOIS, and TMM. His work on recommender systems has received the Best Paper Award Honourable Mention in WWW 2018 and ACM SIGIR 2016. Moreover, he has served as the PC member for several top conferences including SIGIR, WWW, MM, KDD etc., and the regular reviewer for journals including TKDE, TOIS, TMM, TNNLS etc.



Hanwang Zhang is currently an Assistant Professor at Nanyang Technological University, Singapore. He was a research scientist at the Department of Computer Science, Columbia University, USA. He has received the B.Eng (Hons.) degree in computer science from Zhejiang University, Hangzhou, China, in 2009, and the Ph.D. degree in computer science from the National University of Singapore in 2014. His research interest includes computer vision, multimedia, and social media. Dr. Zhang is the recipient of the Best Demo runner-up award in ACM MM 2012, the Best Student Paper award in ACM MM 2013, and the Best Paper Honorable Mention in ACM SIGIR 2016and TOMM best paper award 2018. He is also the winner of Best Ph.D. Thesis Award of School of Computing, National University of Singapore, 2014.