# Learning to Identify New Objects

Yuyin Sun[1], Liefeng Bo[2] and Dieter Fox[1]

***Abstract*— Identifying objects based on language descriptions is an important capability for robots interacting with people in everyday environments. People naturally use attributes and names to refer to objects of interest. Due to the complexity of indoor environments and the fact that people use various ways to refer to objects, a robot frequently encounters new objects or object names. To deal with such situations, a robot must be able to continuously grow its object knowledge base. In this work we introduce a system that organizes objects and names in a semantic hierarchy. Similarity between name words is learned via a hierarchy embedded vector representation. The hierarchy enables reasoning about unknown objects and names. Novel objects are inserted automatically into the knowledge base, where the exact location in the hierarchy is determined by asking a user questions. The questions are informed by the current hierarchy and the appearance of the object. Experiments demonstrate that the learned representation captures the meaning of names and is helpful for object identification with new names.**

## I. INTRODUCTION

Identifying objects specified by a person enables an autonomous robot to do many real world tasks. People refer to objects using natural language describing object *attributes*. For example, "red" is a color attribute and "Chex box" is a name attribute. Attributes have been used in the context of various computer vision tasks, such as object recognition [25], "zero-shot" learning [14] and generating descriptions of unfamiliar objects [5]. Using object names is a powerful way to refer to objects in identification tasks [22]. Previous techniques using attributes assume that all attribute values are known beforehand. However, due to the complexity of indoor environments and since there are many different names that can refer to objects, a robot will inevitablely encounter unknown objects and names. The goal of our work is to enable robots to identify and learn object names in such a challenging setting.

Object names are semantically related and naturally organized in a hierarchy, as illustrated in Fig. 1. Making use of the semantic structure has been shown to outperform building classifiers for each name independently in various learning problems, such as learning many object categories [7], [13] and understanding new words in language [26]. The hierarchical organization of object names is also useful for identifying novel objects. For example, given the name hierarchy shown in Fig. 1, even if a system has never seen a Chex box when a person says "Pick up the Chex" (Fig. 2), the system can identify the correct object using its "Cereal Box" classifier, since "Chex" is known to be a type of cereal.

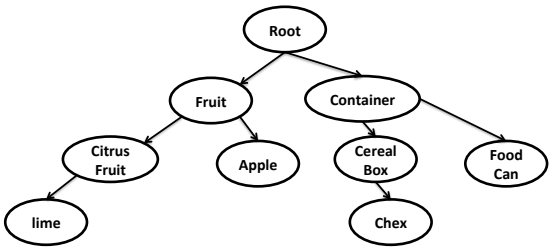



Fig. 1: Part of an object name hierarchy.


Fig. 2: Using the name hierarchy in Fig. 1, the correct object in "Pick up the Chex" can be identified through the "Cereal Box" classifier, even without having observed a Chex box before.

Yet not all names encountered by a robotic system are included in an existing hierarchy. While popular semantic hierarchies such as WordNet [6] cover many category names like "Cereal box", people often refer to objects by their instance names (e.g., "Chex"), which are not included in the hierarchy. People may also use different words for the same concept, e.g., people might call a "mug" a "cup". It is thus important for an autonomous system to automatically extend the hierarchy by inserting new instance names into the right place and associating synonyms with existing words in the hierarchy.

To insert an unknown name into the hierarchy, we need to find a path of words in the hierarchy such that all words are hypernyms of and semantically similar to the target word. A traditional way of measuring the similarity between words is by representing each word using a vector and computing the distance between vectors to find words that are close to each other. Earlier studies represented a word with binary vector representation, of which each element is a bit indicating whether a word exists in a document [15]. More sophisticated semantic vector space models use continuous feature vectors computed from the frequency of context words [24]. This representation has been proven to have good correlation with human judgment of word similarity [8] and is used widely

This work was funded in part by the Intel Science and Technology Center for Pervasive Computing and by ARO grant W911NF-12-1-0197.

[1]Yuyin Sun and Dieter Fox are with the Department of Computer Science & Engineering, University of Washington, Seattle, WA 98115, USA. {sunyuyin, fox}@cs.washington.edu

[2]Liefeng Bo is with ISTC-Pervasive Computing Intel Labs, Seattle, WA 98115, USA liefeng.bo@intel.com

in practice.

In this paper, we propose a method to learn a vector space representation of words which captures hierarchy information. In the new space the distance between words is coherent with the distance within the name hierarchy, such that the system can find a path containing hypernyms of the new word from the existing hierarchy more accurately. We also propose to interact with a user to find the exact meaning of a name in order to extend the name hierarchy.

This paper is organized as follows. After discussing related work in Section II, we introduce the identification system in Section III. We then propose a name learning method in Section IV followed by Section V discussing how to interact with a user for finding the exact meaning of names. We show the effectiveness and efficiency of the proposed system in Section VI. Finally we conclude in Section VII.

## II. RELATED WORK

### A. Learning Attributes

Visual attributes have been successfully used in many computer vision applications. Farhadi et al. [5] describe unfamiliar objects by their attributes and show that attributes are generalizable to new categories. Lampert et al. [14] use attributes for doing "zero-shot" learning and show attributes are useful to detect unseen object categories. Parikh et al. [18] model relative attributes and show the advantages over binary attributes in solving tasks such as image retrieval. Sun et al. [22] use attributes to identify objects. Besides appearance attributes, like color, shape and material, names are also treated as attributes. Most attribute works [5][14][18][22] assume that all attribute values are known beforehand in order to train corresponding classifiers. In practice, however, a robot is likely to face new names since people may use various names to refer to both known and unknown objects.

### B. Learning Word Meaning

Many efforts have been made to learn the meaning of new words. One way is using domain experts' knowledge. A popular example is the WordNet database [6] that organizes words in a hierarchy based on semantic meaning of words. It relates words by synonymy and hypernymy relationship. This relationship is useful for understanding words since if a word is included in WordNet, its meaning can be inferred from its ancestors in the word hierarchy. WordNet covers a large amount of words which are mostly category names.

Another direction is to learn the meaning of words from documents. A word can be represented as a vector [4] generated from a large corpus of documents. The vector is computed based on the word frequency within the context of the target word, where context could be documents, paragraphs or other small windows within documents. This vector representation has been used to solve many useful natural language applications such as search query expansion [10], information retrieval [19] and automatic annotation of text [20]. However, it does not consider structured semantic meaning such as synonym or hypernym relationship.

| Attribute Types | Attribute Values |
|---|---|
| Color | black, blue, brown, gray, green, orange, pink, red, transparent, white, yellow |
| Shape | bag, bowl, circular, cuboid, cylinder, ellipsoid, rectangle |
| Material | ceramic, foam, glass, metal, natural, paper, plastic |

TABLE I: Appearance attribute values used by our system.

Here, we want to learn a vector representation embedding the structured relationship and make the representation coherent with experts' knowledge such as WordNet.

In the computer vision community, many efforts have been made to ground the meaning of words using perceptual information. ImageNet [3] is a large-scale ontology of images built upon WordNet. Each category word is grounded with many images of that category. Matuszek et al. [16] combine perception with language to ground attribute words. Their system focuses on grounding words into color and shape attributes. Lai et al. [13] organize objects as a semantic tree for learning many categories. They assume the system knows all object names and is not able to handle new objects and names. In this work we focus on understanding new names, to be more specific, inserting new names into the right place in an existing hierarchy using an embedding vector space representation of words.

## III. IDENTIFICATION SYSTEM

In this paper, our goal is to build an identification system for RGB-D scenes, so that a human can interact with the system by natural language. In the training stage, we learn a set of attribute classifiers and name classifiers from segmented RGB-D objects and the corresponding labels. In the test stage, a user refers to a target object using natural language describing object attributes and names, for example "Give me my coffee mug. It is blue". Given a scene with $N$ objects $\{o_1, \cdots, o_N\}$ and a sentence, the system first extracts appearance attribute values $\{a^1, \cdots, a^K\}$ and a name $w$ from the sentence, and then identifies the target object by

$$o^* = \arg \max_{o \in \{o_1, \cdots, o_N\}} P(w|o) \ \Pi_{k=1}^{K} P(a^k|o) \qquad (1)$$

where $K$ is the number of attributes used in the sentence. Here, we assume that attributes and name are independent given an object $o$. Next we will discuss how to estimate the likelihoods $P$.

### A. Appearance Attributes

Our system handles 3 types of appearance attributes: color, shape, and material, as shown in Table I. For each attribute type we model the probability using a multinomial logistic regression model as

$$P(a_t^k|o) = \frac{\exp(F_t^k I_o))}{\sum_{\tilde{t}=1}^{T} \exp(F_{\tilde{t}}^k I_o)}, \qquad (2)$$

where $T$ is the number of attribute values for the $k$-th attribute type, $F_t^k$ is the parameter vector of the linear discriminative function, and $I_o$ is the RGB-D feature vector of object $o$ extracted using hierarchical matching pursuit,

a feature learning method proposed by Bo et al. [1]. We learn the parameters $\{F_1^k, \cdots, F_T^k\}$ by maximizing the log-likelihood over labeled objects.

### B. Names

We organize object names as a tree hierarchy $\mathcal{H}$. Each node in the tree is a category like "Fruit", "Container" or "Cereal Box". Each node is associated with all synonyms corresponding to that category. For example, "Cereal Box" may be associated with "Food Boxes" and "Cereal Box". For every non-leaf node with more than one direct child, we learn a multinomial classifier over the objects in the different children of the node. Here, we use the same hierarchical matching pursuit features as for attributes.

If a person uses a name $w$ that is already contained in a node $n_w$ of the tree $\mathcal{H}$, then we compute the probability $P(w|o)$ for an object $o$ using the path from the root of the tree to that node. Let $\boldsymbol{p}^w = [n_1^w \rightarrow \cdots n_{(L-1)}^w \rightarrow n_L^w]$ with $n_L^w = n_w$ denote such a path. Then

$$P(w|o) = \prod_{l=1}^{L-1} P_{n_l^w}(n_{l+1}^w|o) \ . \tag{3}$$

Here, each $P_{n_l^w}(n_{l+1}^w|o)$ is the probability of node $n_{l+1}^w$ given the RGB-D feature of object $o$ evaluated via the classifier trained for node $n_l^w$.

If a person uses a name that is new to the system, it is more challenging to find the correct path. We will handle this case in the next section.

## IV. NAME LEARNING

To handle new names, our identification system needs to associate new names with existing ones so as to find a path. Here, we propose a name learning approach. Our system starts with a set of names, organized in a tree-structured name hierarchy $\mathcal{H}$. Each node in $\mathcal{H}$ contains at least one name word $w_i$. Given a new name $w$, the goal is to find a name path starting from the root node, such that: 1) if the new word is a synonym of an existing word, the new word will be added to the node of its synonym; 2) if the new word is a hyponym (specialization) of a leaf node, then a new node is added as a child of that leaf node and the name is associated with that node. In this work, we assume that the system knows all object categories, therefore don't consider adding new categories.

### A. New Name Inference

Motivated by the success of Socher et al.'s previous work [21], we represent a new name word $w$ as a $d$-dimensional vector $\boldsymbol{v}^w \in \mathbb{R}^d$ using a vector space model. $\boldsymbol{v}^w$ is given by the co-occurence with other words within a large document corpus [21]. Each node $n$ in $\mathcal{H}$ is associated with $M$ names represented as $\{\boldsymbol{v}^{n_w^1}, \cdots, \boldsymbol{v}^{n_w^M}\}$. We use the average $\boldsymbol{v}^n = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{v}^{n_w^m}$ as the vector representation for that name node. Given a word $w$ and a hierarchy $\mathcal{H}$, we want to infer the path $\boldsymbol{p} = [n_1 \rightarrow \cdots n_l \cdots \rightarrow n_L]$ that leads to a node $n_L$ with words that correspond to $w$. Specifically, we define a decision function $\psi : (w, \mathcal{H}, \boldsymbol{p}) \rightarrow \mathbb{R}$ to measure the match score between a path $\boldsymbol{p}$ and a word $w$ given the hierarchy $\mathcal{H}$, such that

$$\boldsymbol{p}^w = \arg\max_{\boldsymbol{p}} \psi(w, \mathcal{H}, \boldsymbol{p}), \tag{4}$$

where $\boldsymbol{p}^w$ is the correct path. In the following, we replace a word with its vector representation $\boldsymbol{v}^w$, and omit $w$ for simplicity. For $\boldsymbol{v}^{n_l}$, we use $\boldsymbol{v}_l$ when there is no confusion.

For a path $\boldsymbol{p}$ with $L$ words, we compute the decision score of the path as the sum of scores of all nodes in it:

$$\psi(\boldsymbol{v}, \mathcal{H}, \boldsymbol{p}) = \sum_{l=1}^{L} \frac{\phi(\boldsymbol{v}, \boldsymbol{v}_l)}{L}, \tag{5}$$

where $\phi(\boldsymbol{v}, \boldsymbol{v}_l)$ measures the match score between words $\boldsymbol{v}$ and $\boldsymbol{v}_l$. The intuition behind (5) is that if $\boldsymbol{p}$ is the real path, $\boldsymbol{v}$ is semantically close to all $\boldsymbol{v}_l$ within the path, resulting in a high score for each $\phi(\boldsymbol{v}, \boldsymbol{v}_l)$. The resulting total score $\psi(\boldsymbol{v}, \mathcal{H}, \boldsymbol{p})$ for the path will then be high as well. The length of the path $L$ is used to normalize the score so as to remove the bias toward long paths.

A natural choice of $\phi(\boldsymbol{v}, \boldsymbol{v}_l)$ is the negative distance between $\boldsymbol{v}$ and $\boldsymbol{v}_l$ as

$$\phi(\boldsymbol{v}, \boldsymbol{v}_l) = -(\boldsymbol{v} - \boldsymbol{v}_l)^T(\boldsymbol{v} - \boldsymbol{v}_l).$$

However, the original vector space does not explicitly consider the structural semantic information embedded in the name hierarchy. The correct path might thus not maximize the score function (5).

### B. New Name Learning

To embed hierarchy information into the score function, we parameterize the score function as

$$\psi_A(\boldsymbol{v}, \mathcal{H}, \boldsymbol{p}) = \sum_{l=1}^{L} \frac{\phi_A(\boldsymbol{v}, \boldsymbol{v}_l)}{L}, \tag{6}$$

where

$$\phi_A(\boldsymbol{v}, \boldsymbol{v}_l) = -(\boldsymbol{v} - \boldsymbol{v}_l)^T A^T A(\boldsymbol{v} - \boldsymbol{v}_l). \tag{7}$$

Here $A \in \mathbb{R}^{d \times d}$ is a linear transformation mapping $\boldsymbol{v}$ into a new space. $A^T\boldsymbol{v}$ is a new representation of word $w$. $\phi_A(\boldsymbol{v}, \boldsymbol{v}_l)$ measures the negative distance between words in the new vector space. Let $W = A^T A$, We can rewrite (7) as:

$$\phi_W(\boldsymbol{v}, \boldsymbol{v}_l) = -(\boldsymbol{v} - \boldsymbol{v}_l)^T W(\boldsymbol{v} - \boldsymbol{v}_l), \tag{8}$$

$W$ uniquely defines the new space and is usually called metric.

Learning the parameterized score function can be formulated as a structure prediction problem [23]. To be consistent with the hierarchy information, the parameterized score function has to satisfy the following constraints:

$$\psi_W(\boldsymbol{v}, \mathcal{H}, \boldsymbol{p}^w) \geq \psi_W(\boldsymbol{v}, \mathcal{H}, \boldsymbol{p}), \forall \boldsymbol{v}, \boldsymbol{p}$$

where $\boldsymbol{p}^w$ is the ground truth path for $\boldsymbol{v}$ provided for the training data. Thus, the correct path should have a higher score than any other path $\boldsymbol{p}$. We define a shorthand

$\delta_W \psi(\boldsymbol{v}, \boldsymbol{p}) \equiv \psi_W(\boldsymbol{v}, \mathcal{H}, \boldsymbol{p}^w) - \psi_W(\boldsymbol{v}, \mathcal{H}, \boldsymbol{p})$, giving us the set of constraints as

$$\delta_W \psi(\boldsymbol{v}, \boldsymbol{p}) \geq 0, \forall \boldsymbol{v}, \boldsymbol{p}. \tag{9}$$

If the set of constraints in (9) is feasible, there typically will be more than one solution $W$. To make the solution unique and avoid overfitting [23], we generalize the idea of max margin and select a low ranking $W$ by minimizing its trace:

$$\begin{aligned} \min_W \quad & \text{tr}(W) \\ \text{s.t.} \quad & \delta_W \psi(\boldsymbol{v}, \boldsymbol{p}) \geq 1, \forall \boldsymbol{v}, \boldsymbol{p} \\ & W \succeq 0. \end{aligned}$$

The condition $W \succeq 0$ enforces $W$ to be Positive Semi-Definite (PSD) so as to be a valid metric and decomposable as $A^T A$.

Different predictions, or paths, $\boldsymbol{p}$ for $\boldsymbol{p}^w$ should be penalized with different loss: if two paths diverge at the higher level of the tree, the loss should be high since two concepts are too far away from each other. We need a loss function $\Delta(\boldsymbol{p}^w, \boldsymbol{p})$ to measure the loss of mis-predicting the real path $\boldsymbol{p}^w$ as $\boldsymbol{p}$ satisfying $\Delta(\boldsymbol{p}^w, \boldsymbol{p}) > 0$ for $\boldsymbol{p} \neq \boldsymbol{p}^w$ and $\Delta(\boldsymbol{p}, \boldsymbol{p}) = 0$. We use the tree loss, which is the deepest level in a tree such that $\boldsymbol{p}$ and $\boldsymbol{p}^w$ are the same, to capture the mis-prediction error. Using this loss, the constraint set becomes

$$\delta_W \psi(\boldsymbol{v}, \boldsymbol{p}) \geq \Delta(\boldsymbol{p}^w, \boldsymbol{p}), \forall \boldsymbol{v}, \boldsymbol{p}$$

We determine the parameter of the path score function via the following convex optimization problem:

$$\begin{aligned} \min_W \quad & \text{tr}(W) + C \sum_{i=1}^{N} \xi_i, \\ \text{s.t.} \quad & \delta_W \psi(\boldsymbol{v}^i, \boldsymbol{p}) \geq \Delta(\boldsymbol{p}^i, \boldsymbol{p}) - \xi_i, \forall i, \boldsymbol{p} \\ & W \succeq 0, \xi_i \geq 0 \end{aligned} \tag{10}$$

where $\xi_i$ are slack variables and $i$ indexes all training examples.

We gather training examples from the known hierarchy $\mathcal{H}$. For each non-root word in $\mathcal{H}$, we can find a unique path from the root. The pair of a node and its path is one training example. In total, we collect $N = |\mathcal{H}| - 1$ training pairs from the hierarchy.

### C. Optimization

The number of constraints in (10) grows quadratically as $|\mathcal{H}|$ increases. Efficiency will thus be an important issue to handle the large set of constraints. Fortunately, only part of the constraints define the feasible set of solutions for the constraint optimization problem, and we can use the cutting-plane method [9] which efficiently solves constraint problems because it uses only active constraints.

We adapt the 1-Slack cutting plane algorithm used by McFee and Lanckriet [17] to our problem. In the original optimization problem (10), there are $N$-Slack variables. The 1-Slack method combines a batch of active constraints specified as $(\boldsymbol{p}_b^1, \cdots, \boldsymbol{p}_b^N)$ into one single constraint. All new constraints share the same slack variable $\xi$. It thereby reduces the number of constraints and slack variables and makes very large optimization problems solvable.

**Input**: Words representation $\{\boldsymbol{v}^1, \cdots, \boldsymbol{v}^N\}$; rankings $\{\boldsymbol{p}^1, \cdots, \boldsymbol{p}^N\}$; Slack trade-off $C$; Convergence tolerance $\epsilon$
**Output**: Metric $W$
1: $\mathcal{W} \leftarrow \emptyset, \xi \leftarrow 0$
2: **do**
3: update metric

$$W = \arg\min_W \ \text{tr}(W) + C\xi$$
$$\text{s.t.} \ \frac{1}{N} \sum_{i=1}^{N} \delta_W \psi(\boldsymbol{v}^i, \boldsymbol{p}_b^i) \geq \frac{1}{N} \sum_{i=1}^{N} \Delta(\boldsymbol{p}^i, \boldsymbol{p}_b^i) - \xi \tag{11}$$
$$\forall \ (\boldsymbol{p}_b^1, \cdots, \boldsymbol{p}_b^N) \in \mathcal{W}$$

4: **for** $1 \rightarrow$ N **do**
5: $\boldsymbol{p}_b^i = \arg\max_{\boldsymbol{p}} \Delta(\boldsymbol{p}^i, \boldsymbol{p}) - \delta_W \psi(\boldsymbol{v}^i, \boldsymbol{p})$
6: **end for**
7: $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\boldsymbol{p}_b^1, \cdots, \boldsymbol{p}_b^N)\}$
8: **while** $\epsilon + \frac{1}{N} \sum_{i=1}^{N} \delta_W \psi(\boldsymbol{v}^i, \boldsymbol{p}_b^i) \geq \frac{1}{N} \sum_{i=1}^{N} \Delta(\boldsymbol{p}^i, \boldsymbol{p}_b^i) - \xi$

**Algorithm** 1: Algorithm for learning hierarchy embedded vector space.

The algorithm is summarized in Algorithm 1. The algorithm alternates between updating $W$ and updating the active constraint set $\mathcal{W}$. Line 3 of Algorithm 1 updates $W$ with constraints specified by $\mathcal{W}$ by gradient descent. The gradient of the function (10) over $W$ is computed as:

$$\frac{\partial f}{\partial W} = I - \frac{C}{N} \sum_{i=1}^{N} \delta_W \psi(\boldsymbol{v}^i, \boldsymbol{p}_*^i),$$

where $*$ indicates the single batch of constraints $(\boldsymbol{p}_b^1, \cdots, \boldsymbol{p}_b^N)$ with the largest gap between $\frac{1}{N} \sum_{i=1}^{N} \psi(\boldsymbol{v}^i, \boldsymbol{p}_b^i)$ and $\frac{1}{N} \sum_{i=1}^{N} \Delta(\boldsymbol{p}^i, \boldsymbol{p}_b^i)$. After the gradient descent update of $W$, the algorithm projects $W$ back onto the set of PSD matrices by spectral decomposition.

After updating $W$, the algorithm finds the $\boldsymbol{p}_b^i$ maximizing the gap between $\Delta(\boldsymbol{p}^i, \boldsymbol{p}^i)$ and $\delta_W \psi(\boldsymbol{v}^i, \boldsymbol{p}^i)$ for the $i$-th word (Line 5). $(\boldsymbol{p}_b^1, \cdots, \boldsymbol{p}_b^N)$ are put together as the $b$-th new batch, which will be added into the active constraint set $\mathcal{W}$.

The cutting-plane method converges when the gap between $\frac{1}{N} \sum_{i=1}^{N} \psi(\boldsymbol{v}^i, \boldsymbol{p}_b^i)$ and $\frac{1}{N} \sum_{i=1}^{N} \Delta(\boldsymbol{p}^i, \boldsymbol{p}_b^i)$ is smaller than a given threshold $\epsilon$ (line 8). Theoretical results guarantee that it converges quickly [9]. In practice (see Section VI) we also found that it converges after only few iterations. Algorithm 1 returns the metric $W$ parametrizing the path score function given as input.

## V. ASKING IDENTIFICATION QUESTIONS

During application of the system, the path $\boldsymbol{p} = [n_1 \rightarrow \cdots n_L]$ determined for a new word $w$ is not always correct. Inserting $w$ into the hierarchy solely based on the most likely path $\boldsymbol{p}$ would introduce substantial noise. To avoid this, our system interacts with a human by asking questions to find the correct place for $w$. The best place here means either

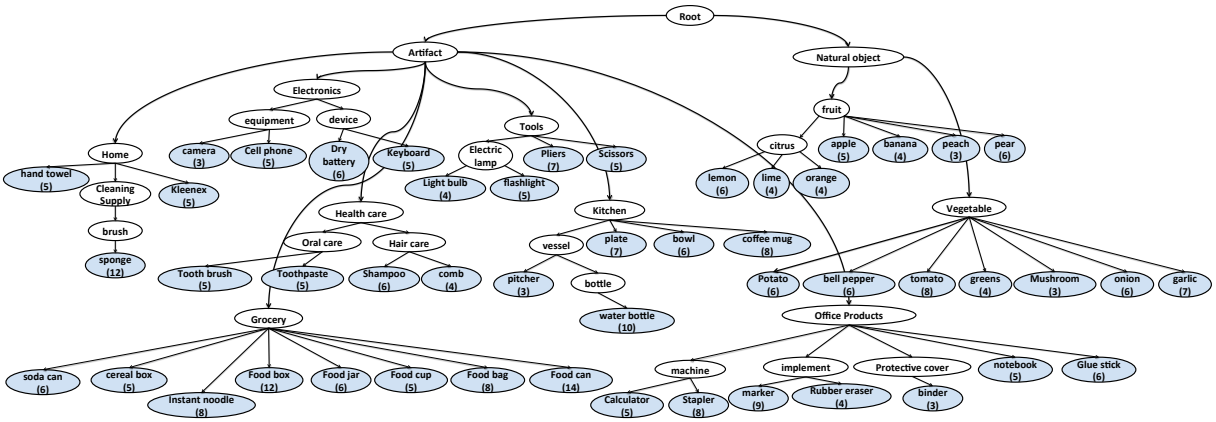



Fig. 3: Hierarchy used in this paper. The numbers in parentheses represent the number of different object instances within each leaf node.

$w$ is a synonym of $n_L$ or $w$ is a hyponym of $n_L$ and $n_L$ is a leaf node in $\mathcal{H}$. The second case indicates $w$ is a new instance name.

### A. Interaction with Humans

Our system interacts with people by asking questions. After finding the most likely path $\boldsymbol{p} = [n_1 \rightarrow \cdots n_L]$, the system asks a question for the last node $n_L$ to determine its relationship with the new word. There are three possible relationships between $w$ and $n_L$: $w$ is a synonym, hyponym or neither of both. If $w$ is a synonym of $n_L$, it will be associated with the corresponding node in the tree. If $w$ is a hyponym of a leaf node $n_L$, $w$ will be inserted into the hierarchy as a new instance name of $n_L$. If $n_L$ is neither case of $w$, $n_L$ should not be part of the correct path $\boldsymbol{p}^w$. The system then finds the next best path and repeats the same procedure. The correct path will eventually be found since we assume a name is either a synonym of an existing word or a new instance name.

### B. Incorporating Perception

In a real setting, a user uses an object name together with several appearance attributes in order to uniquely specify the target object. An intuitive way of deciding which node to ask about could thus be based on information gain, using the posterior uncertainty over the objects in the scene, i.e.

$$n^* = \arg\min_n H[P(o|n, a^1, \cdots, a^k)], \tag{12}$$

where $H[\cdot]$ is the entropy. Unfortunately, due to the hierarchical structure of the tree, (12) cannot be estimated robustly. Our system estimates $P(n, a^1, \cdots, a^k|o)$ instead. Since a person attempts to uniquely specify an object in the scene, we know that only the target object has a high overall score of

$$P(w|o)\Pi_{k=1}^{K} P(a^k|o), \tag{13}$$

while the rest of the objects have far lower scores. Base on this fact, we can infer the relationship between $n_L$ and $w$: we replace $w$ in (13) with $n_L$ to compute the score of (13) for each object in the scene. If no object has a high score, the system infers that $w$ is neither a synonym nor a hyponym for $n_L$ and rejects the hypothesis $n_L$ without asking a question. If there are several objects with high scores, it implies that $n_L$ cannot make the target object unique, which implies that $w$ might be a hyponym of $n_L$, where several objects of the type corresponding to $n_L$ are in the scene. We use a threshold on the score to distinguish these cases.

(a) A scene given to student 1. The student says "Pick up the yellow clipper."

(b) A scene given to student 2. The student says "Pick up the Pepsi."

Fig. 4: Two example scenes.

## VI. EXPERIMENTS

We evaluated our name learning and object identification system with three experiments on our new object attribute dataset. Our results suggest that, (A), name learning significantly outperforms baselines, in terms of accuracy of associating new names with the existing name hierarchy (Section VI-B); (B), name learning significantly boosts the object identification system accuracy, particularly when new names are being used (Section VI-C); and, (C), name learning significantly reduces the human effort required to correctly insert new names into the existing identification system (Section VI-D).

### A. Data Set

We use all 300 objects from the RGB-D object dataset [12] in our experiments. 150 objects were used to train attribute and name classifiers. We demonstrate our system is able to generalize to new instances by testing on all 300 objects. Name and appearance attribute labels for these objects were collected using Amazon Mechanical Turk. These attribute labels are available online at our project website[1]. Fig. 3

[1] http://www.cs.washington.edu/rgbd-object-attributes-dataset/

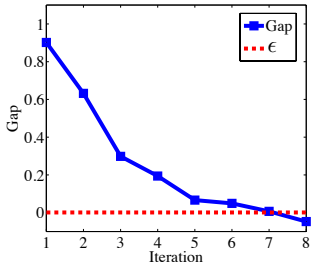



Fig. 5: Convergence rate of the cutting plane method.

shows the full name hierarchy used by our identification system. Leaf names are in blue, with number of object instances in each leaf node given in parantheses. We extracted features to represent object images using Multi-Path Hierarchical Matching Pursuit, a feature learning approach proposed in [1]. Using these features, our system learned attribute classifiers for color, shape, and material, and an object name classifier for each node in the hierarchy using multinomial logistic regression implemented in LibSVM [2].

We generated 600 "scenes", each containing 6 randomly picked objects arranged in image panels. On average, half of the objects have never appeared in the training set. For each scene, we marked one target object by a red rectangle. Two example scenes are shown in Fig. 4. The yellow "pliers" is the target object in Fig. 4 (a), and the "Pepsi" can is the target object in Fig. 4 (b). We asked 2 students at the University of Washington to identify the target object using one sentence for each scene. Each student took charge of 300 scenes.

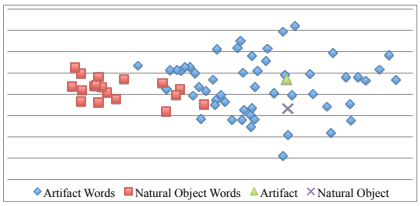
### B. Name Learning

We first examined the convergence rate of the cutting-plane method for learning the tree-based word representation. For a name hierarchy with 72 words, there are $O(72^2)$ constraints in total. Fig. 5 shows that the proposed cutting-plane algorithm converges to the given tolerance $\epsilon$ (set to be $1e^{-5}$ in our experiment) after only 8 iterations. This indicates that our learning technique is highly efficient and can be scaled to even larger data sets.

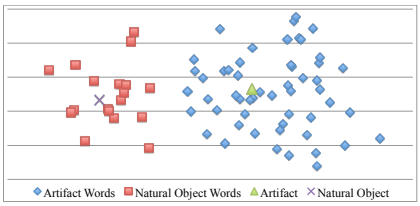
We next illustrate that the proposed name learning method captures name hierarchy information. Recall that each word is originally represented by a vector $\boldsymbol{v}$ extracted from a large corpus of web documents. Fig. 6(a) shows the 2D projection (via PCA) of the original 72 word vectors. All words belonging to the class of natural objects are plotted as red squares and words belonging to the class of artifacts as blue diamonds. The words "Artifact" and "Natural Object" are highlighted using a purple cross and a green triangle, so that one can see how other words are related to their ancestors using the original feature representation.

Our approach learns a metric $W$ capturing hierarchy information. After learning, each $\boldsymbol{v}$ is transformed to a new vector $A\boldsymbol{v}$, where $A$ is found via Cholesky factorization ($W = A^T A$). The learned feature vectors are visualized in Fig. 6(b).

As shown in Fig. 6(a), Artifact words and Natural Object Words are not well separated, and the word "Natural Object" is far away from other Natural Object words. This implies that using the original representation will result in various mistakes when reasoning about novel words, even at the highest level of the hierarchy. Yet in Fig. 6 (b) the two different subcategories are separated from each other and within each category, the name of the category is located right at the center of all words of the same category. This enables our approach to identify nodes for new words more accurately.




(a) Word vector representation without name learning.




(b) Word vector representation after name learning.

Fig. 6: 2-D projection of all 72 word vectors.

| Methods | | IGNORE | VEC_ORI | VEC_HIE |
|---|---|---|---|---|
| Accuracy(%) | Student 1 | 70.67 | 68.00 | **73.67** |
| | Student 2 | 63.67 | 66.70 | **75.00** |
| | AVE | 67.17 | 67.35 | **74.34** |

TABLE II: Identification accuracy on 600 scenes with 6 objects.

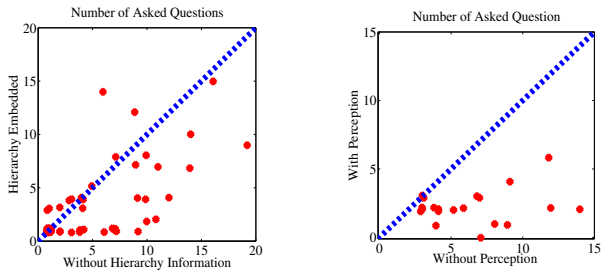
### C. Object Identification

Since our system can associate an unknown name with an existing name in the hierarchy, it can do identification based on new object names. We tested if understanding a new name boosts the identification accuracy.

Each identification task included a scene and a command, such as the example in Fig. 4 (a). We first extracted name and appearance attributes using the Stanford parser [11]. For the name, we used the learned models to determine a name path $\boldsymbol{p} = (w_{p_1}, \cdots, w_{p_L})$ and computed scores of the objects in the scene using that name path. The system estimated the target object using Eq.(1).

We compared the identification accuracy under 3 different settings: 1) An unknown name used to describe the target object was ignored. Only appearance attributes were used for identification (IGNORE); 2) a new name was mapped to a name path (VEC_ORI) using the original vector representation; 3) a new name was mapped using our hierarchy adapted word vectors (VEC_HIE).

The results are summarized in Table II. As can be seen, using the original word vectors to map new names did not achieve higher identification accuracy than simply ignoring unknown names. Adapting the word vectors to the name




(a) Method1 v.s. Method2 (b) Method2 v.s. Method3

Fig. 7: Scatter plot of number of questions asked until the correct location of an unknown name is determined.

hierarchy, however, significantly improved the identification accuracy by a margin of 7% on average. When we evaluated only identification tasks in which a new object name was used, this improvement was about 11%. Overall, our system achieved 74% accuracy using name learning, which is remarkable since 50% of the objects in each scene were not in the training set, including many cases in which even the object of interest and the name used by the person were novel.

As a side note, the improvements for Student 2 were more significant than for Student 1. An inspection of the annotations revealed that Student 2 used more names than appearance attributes to refer to objects, thereby providing more opportunities for our hierarchy to help with the identification task.

### *D. Locating New Names via Questions*

Our name learning method can map a new name into an existing node in the name hierarchy. Yet it still generates various false mappings. Here, we evaluate the performance of our approach to interact with a human by asking clarification questions. Each unique name used by the two students was used for testing on a hierarchy that did not contain that name. As evaluation criterion, we count the number of questions asked to find the correct path for the new name. Three methods were compared: Both Method1 and Method2 were described in Section V-A; Method1 used the original word vector representation while Method2 used the learned vector representation. Method3 was the same as Method2 with extra perception information, as described in Section V-B.

The comparison between Method1 and Method2 is shown in the scatter plot in Fig. 7 (a). For most names, the identification system with learned representation asked fewer questions than the one without name learning. This again suggests that our name learning captures important hierarchy information and is helpful for understanding new names.

We also checked whether perception information helps to further reduce the number of questions. Fig. 7 (b) shows the scatter plot comparing Method2 and Method3 for hard name tasks, which require more than 1 question by Method2. As can be seen, using perceptual information dramatically decreases the number of necessary questions.

To further shed light on the difference between the 3 methods, we intensively examined which questions each method asked for two hard examples. In Example 1, Student 1 was given the scene in Fig. 4 (a). She said "Give me the yellow clipper". The new name "clipper" was a synonym of "pliers" in this identification context. 10 questions were asked in Method1 in total, while 9 questions in Method2. "food can" was asked in Method2 because clipper was used to open food cans such that they appear in a similar document context frequently, resulting in similar word frequency vectors. Using perception information, Method3 is able to detect that there is no food can, lightbulb, electric lamp or scissors. Only 2 questions were asked before finding out that "clipper" is a synonym for "pliers".

| | Method1 | Method2 | Method3 |
|---|---|---|---|
| 1 | food cup | food can | grocery |
| 2 | grocery | grocery | pliers |
| 3 | artifact | artifact | |
| 4 | vessel | home product | |
| 5 | kitchen supply | lightbulb | |
| 6 | flashlight | electric lamp | |
| 7 | electric lamp | tool | |
| 8 | tool | scissors | |
| 9 | scissors | pliers | |
| 10 | pliers | | |

**Example** 1: Questions asked for the new name *clipper*.

| | Method1 | Method2 | Method3 |
|---|---|---|---|
| 1 | potato | onion | grocery |
| 2 | vegetable | vegetable | instant noodles |
| 3 | natural object | natural object | |
| 4 | kleenex | food can | |
| 5 | home product | grocery | |
| 6 | artifact | instant noodles | |
| 7 | coffee mug | soda can | |
| 8 | kitchen supply | | |
| 9 | instant noodles | | |
| 10 | grocery | | |
| 11 | cereal box | | |
| 12 | food cup | | |
| 13 | food box | | |
| 14 | soda can | | |

**Example** 2: Questions asked for the new name *pepsi*.

The second example scene is shown in Fig. 4 (b). The student given the task said, "Pick up the Pepsi." The new name "Pepsi" is a new instance name of "soda can". Example 2 shows the behavior of the different methods. The identification system asked 14 questions to find out the right path for "Pepsi" using Method1. Method1 asked many unrelated names such as "kleenex" and "home product", while Method2 quickly found out that "Pepsi" is more related to "grocery" by using the hierarchy information. Only 7 questions were asked in this case. Using attribute and object classifiers, the system filtered "onion", "vegetable", "natural object" and "food can" which did not exist in the scene. It also found that "pepsi" is not a synonym of "soda can" because there is still uncertainty in the scene simply using "soda can" to refer to the target. Our system found that "Pepsi" is a new instance of "soda can" after asking 2 questions.

## VII. Conclusion

A robot operating in indoor environments and interacting with people will frequently face new objects and new names used for referring to objects. It therefore has to be able to learn new objects and names as they occur during task execution. Toward this goal, we propose a new object recognition system for identifying objects based on natural language referrals. Based on a hierarchical, semantic organization of objects and their names, our system can identify even unknown objects and when a person uses novel names for these objects. To reason about new object names, our system leverages co-occurrence frequencies of words in large document corpora and learns a new vector representation suitable for our object hierarchy.

Extensive experiments show that our learned word vector representation achieves significantly better reasoning about unknown names than the original word vectors. Our overall system achieves 74% identification accuracy for scenes containing six objects, with half of the objects being unknown on average. This is an extremely challenging task, since not only might the object of interest be novel to the system, but the system doesn't even know whether a new name refers to a novel object or is a synonym for a known object. Our experiments also demonstrate that our approach is able to determine the correct location of an unknown object name in the hierarchy by asking a user only a small number of questions. Results show that combining perceptual information with name reasoning significantly reduces the number of necessary questions.

There are some situations in realistic scenarios that are not addressed by our current system. In the previous discussion, we assume that there is only one target object in the scene. In the real world, a robot may encounter cases where a person refers to none or multiple of the visible objects. For the first case, we can set a decision threshold for identification confidence, such that the system rejects to identify any object because there is no target object. For the latter case with multiple objects, the system may either identify any of the target objects or require more attributes, such as the relative location, to localize a specific object.

Some mistakes in learning names may happen in our current system. For instance, a person may use "Pepsi" to refer to a single "Soda Can" in a scene and our current system may infer "Pepsi" as a synonym of "Soda Can". This kind of failure is not very likely to happen because peopley typically use higher level names ("Soda Can") rather than lower level names ("Pepsi") of an object unless necessary. However, to avoid introducing noise into the hierarchy, a fully automatic name learning system still needs to provide some way to frequently clean up the hierarchy, which is part of future work.

In this paper, the experimental results on artificially created scenes have demonstrated the feasibility of the proposed algorithm for learning new names. In real world scenes, there will be some additional challenges for our system, such as highly cluttered scenes resulting in noisy object segmentation. Finally, our system only adds nodes at the leaves of the hierarchy, assuming that all object classes are given by a repository such as Wordnet. Adding internal nodes is another opportunity for future research.

## References


[1] L. Bo, X. Ren, and D. Fox. Multipath sparse coding using hierarchical matching pursuit. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[2] C. Chang and C. Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011.

[3] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[4] K. Erk. Vector Space Models of Word Meaning and Phrase Meaning: A Survey. *Language and Linguistics Compass*, 2012.

[5] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing Objects by their Attributes. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2009.

[6] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[7] G. Griffin and P. Perona. Learning and using taxonomies for fast visual categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[8] T. Griffiths, J. Tenenbaum, and M. Steyvers. Topics in semantic representation. *Psychological Review*, 2007.

[9] T. Joachims, T. Finley, and C. Yu. Cutting-plane training of structural svms. In *Machine Learning*, 2009.

[10] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *International Conference on World Wide Web*, 2006.

[11] D. Klein and C. Manning. Accurate unlexicalized parsing. In *Association of Computational Linguistic*, 2003.

[12] K. Lai, L. Bo, X. Ren, and D. Fox. A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. In *IEEE International Conference on Robotics and Automation*, 2011.

[13] K. Lai, L. Bo, X. Ren, and D. Fox. A Scalable Tree-based Approach for Joint Object and Pose Recognition. In *the AAAI Conference on Artificial Intelligence*, 2011.

[14] C. Lampert, H. Nickisch, and S. Harmeling. Learning to Detect Unseen Object Classes by Between-Class Attribute Transfer. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2009.

[15] W. Lowe. Towards a theory of semantic space. In *Annual Conference of the Coginitive Science Society*, 2001.

[16] C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox. A Joint Model of Language and Perception for Grounded Attribute Learning. In *International Conference on Machine Learning*, 2012.

[17] B. McFee and G. Lanckriet. Metric learning to rank. In *International Conference on Machine Learning*, 2010.

[18] D. Parikh and K. Grauman. Relative Attributes. In *IEEE International Conference on Computer Vision*, 2011.

[19] M. Pasca, D. Lin, J. Bigham A. Lifchits, and A. Jain. Names and similarities on the web: Fact extraction in the fast lane. In *Association of Computational Linguistic*, 2006.

[20] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *Association of Computational Linguistic*, 2011.

[21] R. Socher, B. Huval, C. Manning, and A. Ng. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *Conference on Empirical Methods in Natural Language Processing*, 2012.

[22] Y. Sun, L. Bo, and D. Fox. Attribute based object identification. In *IEEE International Conference on Robotics and Automation*, 2013.

[23] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *International Conference on Machine Learning*, 2005.

[24] P. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 2010.

[25] J. Wang, K. Markert, and M. Everingham. Learning models for object recognition from natural language descriptions. In *British Machine Vision Conference*, 2009.

[26] F. Xu and J. Tenenbaum. Word learning as bayesian inference. In *Annual Conference of the Cognitive Science Society*, 2000.