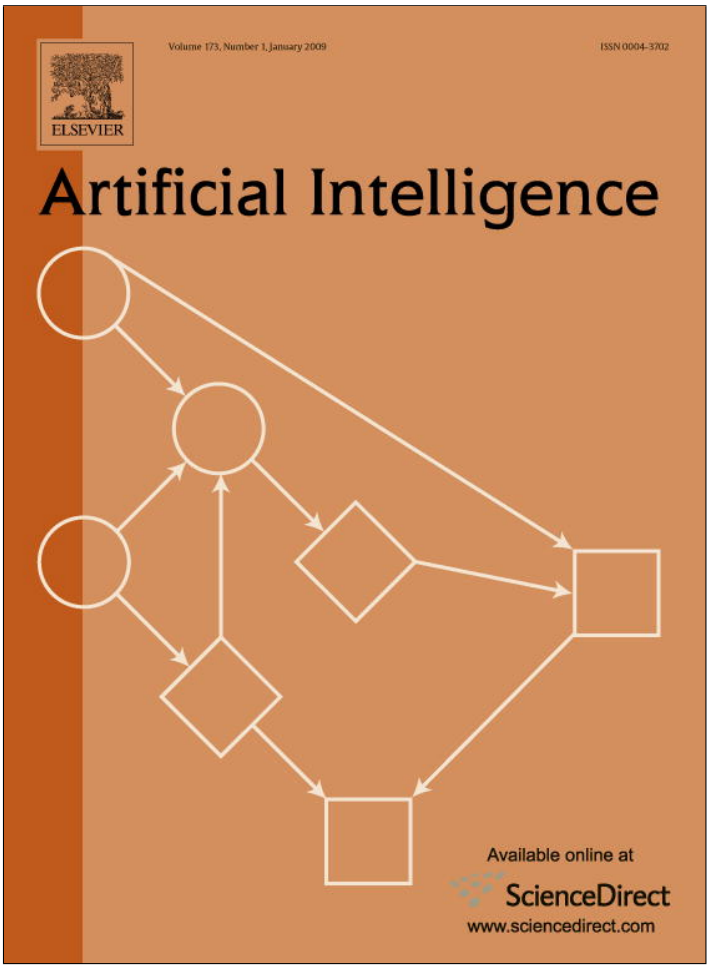

Artificial Intelligence 173 (2009) 145–166

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Robotic vocabulary building using extension inference and implicit contrast

Kevin Gold *, Marek Doniec, Christopher Crick, Brian Scassellati

Department of Computer Science, Yale University, New Haven, CT, USA

ARTICLE INFO

Article history:
Received 24 September 2007
Received in revised form 16 September 2008
Accepted 16 September 2008
Available online 25 September 2008

Keywords:
Word learning
Context
Developmental robotics
Robot
Decision trees
Symbol grounding
TWIG
Pronouns
Autonomous mental development
Inference

ABSTRACT

TWIG ("Transportable Word Intension Generator") is a system that allows a robot to learn compositional meanings for new words that are grounded in its sensory capabilities. The system is novel in its use of logical semantics to infer which entities in the environment are the referents (extensions) of unfamiliar words; its ability to learn the meanings of deictic ("I," "this") pronouns in a real sensory environment; its use of decision trees to implicitly contrast new word definitions with existing ones, thereby creating more complex definitions than if each word were treated as a separate learning problem; and its ability to use words learned in an unsupervised manner in complete grammatical sentences for production, comprehension, or referent inference. In an experiment with a physically embodied robot, TWIG learns grounded meanings for the words "I" and "you," learns that "this" and "that" refer to objects of varying proximity, that "he" is someone talked about in the third person, and that "above" and "below" refer to height differences between objects. Follow-up experiments demonstrate the system's ability to learn different conjugations of "to be"; show that removing either the extension inference or implicit contrast components of the system results in worse definitions; and demonstrate how decision trees can be used to model shifts in meaning based on context in the case of color words.

1. Introduction

Robots that could understand arbitrary sentences in natural language would be incredibly useful—but real vocabularies are huge. American high school graduates have learned, on average, about 60,000 words [29]. Programming a robot to be able to understand each of these words in terms of its own sensors and experience would be a monumental undertaking. To recognize the words in the audio stream is one thing; to recognize their referents in the real world, quite another. The messy, noisy nature of sensory input can render this difficult problem intractable for the programmer. It would be far more useful to have a robot that is able to learn the meanings of new words, grounded in its own perceptual and conceptual capabilities, and ideally with a minimum of user intervention for training and feedback. The present paper describes a system that is built to do exactly that.

According to the developmental psychology literature, children can learn new words rapidly [7] and in a largely unsupervised manner [9]. From the time that word learning begins around the age of one, children's word learning accelerates from a rate of about word every 3 days for the first 4 months [14], to 3.6 words a day between the ages of 2 $\frac{1}{2}$ and 6 [1,4,14], to 12 words a day between the ages of 8 and 10 [1]. This learning would impose quite a demand on the time of parents and educators if children had to be taught all these words explicitly, but instruction does not appear to be necessary for learning

* Corresponding author.
E-mail addresses: kgold@wellesley.edu (K. Gold), doniec@mit.edu (M. Doniec), christopher.crick@yale.edu (C. Crick), scaz@cs.yale.edu (B. Scassellati).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2008.09.002

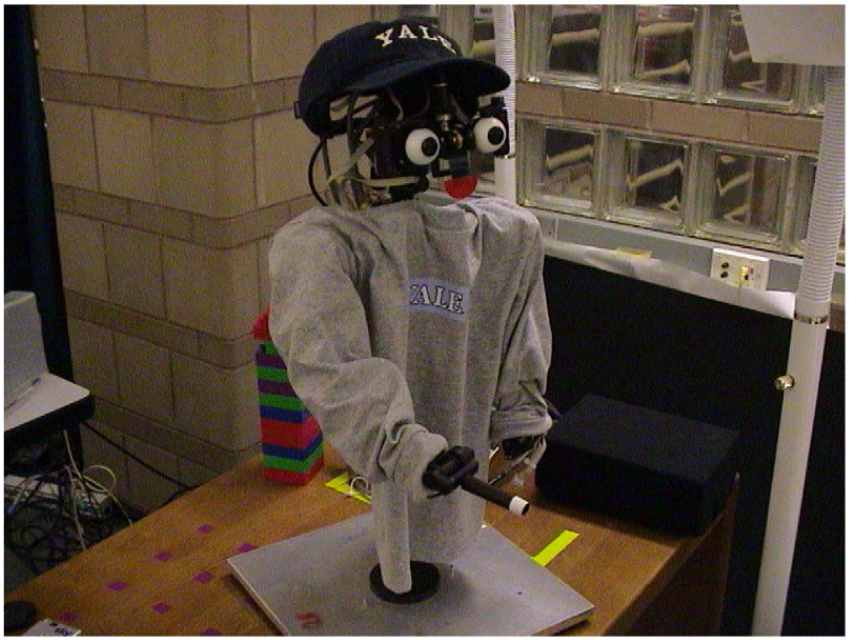

Fig. 1. The robot Nico, on which the TWIG word learning system was implemented.

language [4,27], and in fact there are communities in which children learn to speak with hardly any adult instruction at all [21]. Thus, in designing a robotic system built to learn the meanings of words quickly and without supervision, it makes sense to study the heuristics that children appear to use in learning the meanings of words.

One strategy that children use is to infer what object or person an new word is referring to from its grammatical context—realizing that "sibbing" must refer to a verb, but "a sib" must refer to a noun [6]. Another strategy children use is to contrast the new word with other words they know in order to narrow down its meaning without explicit feedback [7,9,26]. For instance, a child may believe at first that "dog" refers to all four-legged animals, but on learning the word "cat," the child realizes that implicitly, the meaning of "dog" does not include cats, and so the definition of "dog" is narrowed. This paper describes a word learning system called TWIG that employs both of these strategies, using sentence context and implicit contrast to learn the meanings of words from sensory information in the absence of feedback.

Until now, research into learning the semantics of words in an unsupervised fashion has fallen primarily into two camps. On the one hand, there have been attempts to learn word semantics in simulation, with the world represented as collections of atomic symbols [12] or statements in predicate logic [41]. These simulations generally assume that the robot already possesses concepts for each of the words to be learned, and thus word learning becomes a simple problem of matching words to concepts. Such systems can deal with arbitrarily abstract concepts, because they never have to identify them "in the wild." On the other hand, systems that have dealt with real sensor data have generally assumed that concepts and even word boundaries in speech must be learned at the same time as words. Since learning abstract concepts directly from sensor data is difficult, these systems have focused on learning words for physical objects [37] and sometimes physical actions [48].

The TWIG word-learning system is an attempt to bridge this divide. "TWIG" stands for "Transportable Word Intension Generator." It is "transportable" in the sense that it does not rely too heavily on the particular sensory setup of our robot (Fig. 1), but could be moved with ease to any robot that can express its sensory input in Prolog-like predicates that can include numeric values. (Note that it is the learning system, and not the definitions learned, that is transportable; each robot using TWIG will learn definitions specific to its own sensory capabilities.) A "word intension" is a function that, when applied to an object or relation, returns true if the word applies. In other words, an intension is a word meaning [8,13]. Though TWIG was originally designed to solve the conundrums posed by pronouns, which themselves straddle the boundary between the abstract and the concrete, its techniques are more generally applicable to other word categories, including verbs, prepositions, and nouns.

The first technique that TWIG introduces is *extension inference*, in which TWIG infers from sentence context what actual object the new word refers to. For example, if the system hears "Foo got the ball," and it sees Bob holding the ball, it will assume that "Foo" refers to Bob; the entity in the world that is Bob is the *extension* of "Foo" in this example. Doing this allows the system to be picky about what facts it associates with a word, greatly reducing the impact of irrelevant information on learning. Grammatical context also informs the system as to whether it should be looking for facts about a single object or a relation between objects, as with prepositions and transitive verbs.

Second, to build word definitions, the system creates *definition trees* (see Section 4.3). Definition trees can be thought of as reconstructing the speaker's decision process in choosing a word. The interior nodes are simple binary predicates, and the words are stored at the leaves. The meaning of a word can be reconstructed by following a path from the word back to the root; its definition is the conjunction of the true or negated predicates on this path. Alternately, the system can choose

a word for a particular object or relation by starting at the root and following the branches that the object satisfies until arriving at a word.

Definition trees are a type of decision tree, and are created in a manner closely following the ID3 algorithm [32]—but treating word learning as a problem of word *decision*, with words defined in terms of the choices that lead to them, is an approach novel to TWIG. Previous robotic systems have typically assumed that word concepts can overlap with each other arbitrarily [37,48]. In assuming that words do not overlap, TWIG actually allows greater generalization from few examples, because the boundaries of each concept can extend all the way up to the "borders" of the other words.

Though we described an earlier version of the TWIG system in a previous conference paper [17], that version of TWIG did not make use of the definition tree method, which we developed later [15]. (Definition trees were originally simply called "word trees," but this led to some confusion with parse trees.) The latest version of TWIG includes several enhancements that were not present in any previously reported results, including part-of-speech inference, online vocabulary updates that allow new words to be used immediately, and better handling of negated predicates. This paper also presents a new quantitative evaluation of the two halves of the system, and some new experiments that demonstrate the system's flexibility in building on its previous vocabulary and learning multiple definitions for the same word.

2. Problems addressed by TWIG

The TWIG system is best understood in the context of the problems it has been built to address. Word learning might at first seem to be a problem of simply associating sights and sounds, but as we shall see, language is more subtle than this, and many ostensibly reasonable approaches cannot succeed in the long run.

2.1. Finding the referent

One of the reasons that real children are so adept at learning language is that they appear to be able to learn language from conversations which they are not themselves involved in [27]. Unfortunately, this means that many new words may not be clarified with pointing gestures or gaze direction. How, then, is the robot to know what objects to attend to when it hears a new word?

It is tempting to build a robot that simply associates every sound sequence heard with everything in its environment, hoping that statistically, words will appear with their referents more often than not. However, there are logical problems with this approach [4]. One of the most critical is that some words refer to properties that are almost always present somewhere in the environment, making the word uninformative as to the presence or absence of the property. For example, "I" refers to the speaker—but the word "I" is not informative as to whether someone in the environment is speaking, because the fact that any words at all are being spoken implies this. As another example, it will almost always be the case that one thing in the environment is "above" another—so the presence or absence of "aboveness" somewhere in the environment does not vary much with the presence or absence of the word.

Previous word learning systems have largely ignored the problem of finding the referent, either treating it as already solved by a black box [2,34,37] or attempting to associate words with everything in the environment [12]. Chen Yu's eye-tracking system was a notable exception, as it used gaze to determine reference [48]. However, gaze direction can still be ambiguous, particularly for words that describe relations. TWIG uses sentence context to determine reference, as described below.

2.2. Using language to learn more language

Once some lexicon is known, it should become easier to learn other words. If a speaker of Babelese points to a rabbit eating a carrot and says, "Gavagai foo labra," the meaning is highly unclear; but if we know that "gavagai" means *rabbit* and "labra" means *carrot*, then "foo" is quite likely to mean "eat." Going still further, if we know that the most common word for "eat" is actually "barra" and not "foo," we might look for differences between what the rabbit is doing to the carrot now and how it looks to "barra" a carrot; perhaps "foo" actually means *to eat quickly*, or simply *to enjoy*.[1]

Previous systems have employed a weak form of this kind of inference by treating the meaning M of a sentence as an unordered set of symbols and finding a mapping from words to subsets of symbols such that the union of symbol subsets covers M [41]. This means that if one word in a sentence is known to mean "rabbit," it is unlikely that other words in the same sentence will be assigned to "rabbit" as well. This is one way that increased vocabulary size can help learn new words, and it has been successfully employed in the past [12,41,48]. Still, treating sentences and their meanings as unordered sets tends to lose some information; for instance, there is no way to learn the difference between "above" and "below" without taking word order into account, since the order in which the nouns are spoken is the only difference between these prepositions.

The second way in which learning words can become easier as vocabulary size increases is that new words can be contrasted with known words for differences. This is a fairly novel approach for TWIG, and one of its great strengths;

[1] The example of "gavagai" referring to a rabbit in some way is a classic example from Quine [31], though Quine was making a rather different point.

previous systems have generally assumed that words can overlap arbitrarily in meaning, and that they therefore do not inform each others' boundaries in concept space [12,34,37,48]. If one assumes (as many writers do) that no two words mean quite the same thing, then the learner can assume that contrasts between words are informative. In fact, young children appear to follow this heuristic when learning new words, and will reject words that are technically accurate if more precise words are available [26]. Thus, existing definitions are a resource that can and should be exploited when learning new definitions, regardless of whether the old words appear in the same sentence as the new words.

2.3. Negation with no negative examples

The word "he" means roughly "someone who is male, and is neither the addressee nor the speaker." These last conditions may be difficult to learn from passive observation, however, because the observed speakers will typically only use "he" correctly, making the definition indistinguishable from "someone who is male." While it is true that all the learner's examples of "he" will satisfy "not the speaker," there are many other negated predicates that will be coincidentally satisfied every time "he" is spoken—for example, the speaker is not on the moon, it is not the year 3000, the referent is not the King of the United States, and so on. Including the values of all of these predicates as part of the definition of "he" would be incredibly unwieldy, if not impossible in the limit. How is the learner to note the significance of "not speaker," but discount the importance of "not the year 3000"?

This problem of knowing to ignore irrelevant facts is just as true for the case of unnegated facts, but is especially acute for negated facts because there are so many of them that hold at any given time. If there are n disjoint classifications for an object, then $n - 1$ will not apply. Obviously, one could give up on the unsupervised approach and introduce positive and negative feedback to get around this problem, but TWIG uses a more elegant solution: introduce negated clauses only when they are necessary to contrast with other word meanings. In this way, definitions are produced that are not only more accurate, but also more concise and more consistent with Grice's maxims [18]. Again, this approach will be described in more detail below.

2.4. Abandoning finite concept regions

When given only positive examples of a concept, it can seem reasonable to construct a concept hypothesis that consists of only a small, finite region of the sensory space that is centered about those examples. For example, if one's examples of "hot" temperatures are all between 25–30° C, an algorithm might construct a hypothesis that a day is "hot" iff it falls within this range, or perhaps within 23–32 °C just to allow for a margin of error. Another approach might assume that hot temperatures obey a normal distribution about 27.5 °C. Assuming that concepts occupy only finite or highly concentrated regions of the concept space is common in word learning work that deals with real visual input [37,48].

The problem with these approaches is that many word concepts, such as "hot" and "far," actually extend to infinity along their relevant dimensions. When confronted with a 40 °C temperature, the algorithms of the previous example would all classify it as unlikely to be "hot," because the new example is so distant from the old examples. Yet children do not have to encounter infinitely hot items to know that if one keeps increasing a temperature, it just keeps getting hotter.

TWIG addresses this problem by assuming that all known words form a partition of the concept space, and that the relevant decisions can all be treated as thresholds that divide the space. Thus, if TWIG had access to examples of "cold" ranging from 0–5 °C and "hot" ranging from 25–30 °C, it would assume (in the absence of other words) that all temperatures less than 0 °C are "cold" and that all temperatures greater than 30 °C are "hot." Further details can be found under "Definition Trees," below.

2.5. Roles and relations as definitions

The definition of "you" can be stated as, "the person(s) whom the speaker is addressing." No amount of analyzing a person in isolation can determine whether the person counts as a "you"; one must know whether the speaker is addressing the person. Any word learning algorithm that removes the referent entirely from its context would therefore fail to learn the meaning of "you." The same is true for other *deictic* pronouns, or pronouns that encode a relationship to the speaker: "I," "you," and "this," for example. The relations that are encoded in deictic words can vary from language to language; they can include height relative to the speaker (Daga, spoken in Papua New Guinea), whether motion is toward or away from the speaker ("come," "go," various motion morphemes in Somali), whether something is upriver or downriver (Yup'ik, spoken in Alaska), and even whether the speaker came to know something through hearsay or direct evidence (Makah, spoken in Washington State) [39].

TWIG allows relations to the speaker as possible clauses in word definitions, such as "is within 30 cm of the speaker" or "someone the speaker is looking at." This may seem to be a solution particular to deictic pronouns, but taking the speaker's perspective into account is probably more widely applicable as well. For instance, "good" means little in an absolute sense, but implies something about the speaker's evaluation of an object; it implies a relation between the speaker and the referent.

Definitions involving relations between objects, such as those implied by prepositions and transitive verbs, are handled in a very similar way, and thus with little extra overhead.

Table 1
A summary of the most influential semantics learning work to date, compared to TWIG: Siskind's noise-resistant semantics learner [41], de Marcken's word segmentation algorithm with extensions to semantics [12], Regier's preposition-learning neural network [34], Bailey's verb prototype learner [2], the CELL system [37], and Yu and Ballard's eyetracking word learner [48]. Parts of speech: p = preposition, v = verb, n = noun, pn = pronoun.

	Siskind	de Marcken	Regier	Bailey	CELL	Yu	TWIG
Parts of speech	all	all	p	v	n	n, v	n, v, p, pn
Real sensors					+	+	+
Deixis			+				+
Numerical data			+		+	+	+
Compositional meanings	+	+					+
Produces sentences							+
Word segmentation		+			+	+	
Visual prototypes			+		+	+	

2.6. Language generation

Once new words are learned, it should be possible to combine them to form meaningful sentences. Most previous robotic work has focused on the pre-grammatical phase of word learning. The learned words can be picked out of the audio stream, and the best word could be found for a given referent, but this left something to be desired in terms of giving the robot the ability to communicate.

TWIG understands and generates sentences within a framework of formal semantics, using Prolog to understand and generate new sentences. The definitions it creates are predicate calculus expressions that are ready to be combined during parsing or generation into logical forms. While this strategy makes the system somewhat more brittle in terms of recognition, as utterances must match a recognized grammatical parse, it leaves the system in a position to generate correct full sentences about its environment, rather than simply classify referents with single-word utterances.

2.7. Summary of previous systems

Table 1 summarizes some of the differences between TWIG and the most influential recent word learning systems. Though each of these systems is essentially attempting to model the same problem of how children learn the meanings of new words without feedback, they have varied greatly in the amount of abstraction they assume in the data and the kinds of words they attempt to model.

Systems using real sensors, such as Roy and Pentland's CELL system [37], Steels and Kaplan's Aibo word learner [42], and Yu and Ballard's eyetracking word learner [48], have typically been only able to learn definitions for concrete nouns, and in Yu and Ballard's case, some physical verbs. Real sensors typically do not provide the predicates necessary to characterize more abstract words. These systems' word semantics have been largely non-compositional, and therefore difficult to integrate into a language generation system. On the other hand, these systems typically solved the problems of finding word boundaries [37,48] and creating concepts for visual classification [37,42,48], two problems that TWIG does not currently address.

Algorithms that attempt to learn word meanings from text alone, such as "Latent Semantic Analysis" [23], typically make use of the proximity of words in text to create a metric of similarity between words. While such methods have their place in text retrieval, methods that do not make use of extra-textual information are doomed to create definitions that are ultimately circular, and useless for the task of semantics grounded in sensors [35]. Such methods would face substantial hurdles in being adapted to real environments, since a physical environment is not very similar to a text corpus.

Systems built on predicate logic representations, such as Siskind's noise-resistant word learner [41] and Kate and Mooney's KRISPER [22], generally treat the word learning problem as finding a mapping from one set of symbols to another, and do not take into account the fact that real input would probably be vector or scalar in nature. They have generally assumed that the exact predicate or metalanguage symbol necessary to define a word already exists somewhere in the input. This allowed for a great degree of generality, but this generality is somewhat artificial, as all the burden of creating word concepts is shifted to the sensory system. Predicate logic-based word learners extend the farthest back in AI history, since they could be implemented in simulation with a great deal of abstraction; the earliest example is Terry Winograd's SHRDLU system, which could ask for natural language definitions when it encountered unknown words [45]. Few simulations have modeled the impact of sensor noise and recognition error on the learning, though more have modeled referential uncertainty [22,41].

Systems that use more realistic simulated sensory input, such as Regier's preposition-learning neural network [34] and Bailey's verb learner [2], have typically focused on learning a single part of speech in a highly simplified environment that consists of the referents alone. Because these systems were more psychological models than practical implementations, they tended to make optimistic assumptions about the sensors (noise-free, discretely valued) and the environment (empty except for the referents).

TWIG employs a hybrid approach as well, working with predicates generated from real sensors that include some scalar parameters. Predicate logic allows TWIG to maintain a representation of the world that includes relations between objects,

while the ability to handle scalar data and form conjunctions keeps it from placing unreasonable demands on the programmer of the sensory system in providing ready-made concepts. TWIG achieves this balance of linguistic expressivity and robustness to the vagaries of real input through a curious combination of old-fashioned intensional logic (the intension/extension distinction was implemented as far back as LUNAR [46]) and a more modern machine learning approach.

Though the idea of using trees to represent distinctions in meaning has been used before to model word semantics [19] and store word meanings [3], TWIG is the first system to learn tree structures for storing word semantics in an unsupervised fashion.

3. Robotic platform

This section is about the particular robot on which TWIG was implemented, and the predicates it generated from its sensors. Nothing in this section is a part of TWIG proper, but it will be useful when reading the description of TWIG to have an idea of the kinds of input it is meant to process.

TWIG was implemented on Nico, a humanoid non-mobile robot that uses video cameras, dual-channel microphones, and the Cricket indoor location system [30] to sense the world (Fig. 1). Nico is a general research platform, and has been used for experiments in learning to point [43], modeling infant looking-time experiments [25], testing algorithms for intention recognition [10], drumming to a conductor's beat [11], and mirror self-recognition [16]. Thus, the robot's sensors and design are not particularly specific to TWIG. Conversely, the TWIG system is flexible about what kinds of predicates it uses, and in theory it could work with any robotic system that can provide some kind of simple object representation.

On processing an utterance, TWIG queries the robot's sensors to receive a predicate-based description of the robot's world. The predicates used in our experiments, and their generation from Nico's sensors, shall now be described in greater detail.

3.1. Vision: Faces and gaze direction

The robot's visual system was used for finding faces and determining the directions they faced. A wide-angle CCD camera in Nico's right eye grabbed 320×240 images at 30 frames per second. These frames were processed by two separate face detectors using the Viola and Jones object finding algorithm [44], as implemented in OpenCV [5]. One face detector was used to find profile faces, the other, faces looking directly at the robot. Output from these two face detectors was combined using the forward algorithm [38] to give an estimate of whether a person was more likely to be looking to the side or directly at the robot at a given time. The forward algorithm also functioned to reduce the number of erroneous face detections.

For each face detected in the visual scene, a new symbol *personL* or *personR* was added to the logical representation of the environment, corresponding to whether the person was detected on the left or ride side of the visual field. (This representation was somewhat tailored to the experiment, but made the data more human-readable.) For each of these symbols, `person(X)` was added to the environment. In addition, `lookingAt(X,Y)` was true for any pair X, Y such that Y was in the half-space 30 cm away from X, on the other side of the plane to which X's looking direction was perpendicular.

The robot also used a symbol for itself, *nico*, and assumed `person(`*nico*`)`. When someone else was speaking, `lookingAt(`*nico*`,Y)` was true for any Y in the robot's visual field. During language generation, `lookingAt(`*nico*`,Y)` was set to the person whom the robot was addressing.

3.2. Sensor networks: Object localization and distance

To simplify the perceptual problem of finding objects and distances between them, the robot used the Cricket Indoor Location System [30]. Two objects, a tennis ball and a plush pig toy were equipped with Cricket receivers (Fig. 2), while the laboratory ceiling was equipped with 9 Cricket beacons arranged in a 3×3 grid, each roughly 1.5 m from its neighbors. The beacons and receivers communicated via ultrasound signals to determine distances between each, and the robot performed a least-squares calculation to determine the location of each object in 3-dimensional space.

For each sensor-object, a symbol *obj1* or *obj2* was added to the environment. The distance in centimeters between each entity in the environment was then calculated and added to the environment with the `dist(X,Y,V)` predicate, e.g., `dist(`*perL, obj1,* `30.5)`. For the purpose of this calculation, the faces described earlier were assumed to be a fixed distance of 60 cm from the robot, since the vision system did not have access to visual depth information.

In addition, the absolute height above the ground for each object could be computed from the Cricket sensors. The difference `height(X) - height(Y)` for each object pair (X, Y) was encoded as the predicate `relHeight(X,Y,V)`.

3.3. Identity predicates

In addition to the sensory predicates described above, each object received a predicate that was uniquely true of itself, such as `obj1(`*obj1*`)`. This was to allow for the possibility of the robot mistakenly believing that one of the words to be learned was a proper noun. In addition, each object satisfied the identity relation `ident(X,X)` with itself.

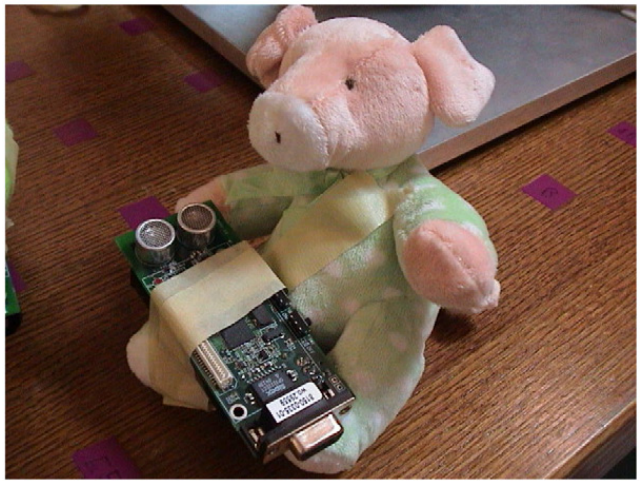

Fig. 2. The pig toy used in the experiment, attached to the Cricket sensor the robot used to find it.

3.4. Audio: Speaker detection and speech segmentation

A dual-channel microphone was used to determine the speaker of each utterance. The two microphone heads were placed 30 cm apart and 50 cm in front of the robot. Within the Sphinx-4 speech recognition system, whether the speaker was to the left or right was determined by comparing the average volume of the two channels over time. The symbol for the corresponding person, `perL` or `perR`, was then bound to the variable *S* (for speaker).

For the purpose of speech recognition, Sphinx used a context-free grammar:

```
<s> = <np> <vp>;
<np> = (this | that| the ball | the pig | i | you | he);
<vp> = (is <np> | is <p> <np> | got <np>);
<p> = (near | above | below);
```

Since Sphinx was not trained for the particular speakers in our experiments, even this small grammar resulted in a fairly high recognition error rate. A speaker-dependent speech system would have allowed a larger vocabulary of recognized words.

3.5. Environment summary

When it begins to parse an utterance, TWIG queries the robot for the state of the world, which the robot produces in the form of a list of all atomic sentences that the robot can produce from its available sensors at the time of query. In the experiments to be described, this list looked something like this:

```
% Unique predicates -- obj1/obj2 = ball & pig,
% perL/perR = person on left/right
obj1(obj1), obj2(obj2),
perL(perL), perR(perR),
% "Person" predicates (faces)
person(perL),
person(perR),
% Possible facing directions
lookingAt(perL, perR),
lookingAt(perL, obj1),
...
% Identity relations
ident(obj1, obj1),
ident(obj2, obj2),
...
% Distances
dist(perL,obj1,31.568943),
dist(perL,obj2,59.623984),
...
```

```
% Relative height
relHeight(obj1, obj2, 40.23904)
relHeight(obj1, perL, -5.239)
...
```

TWIG's parsing and learning mechanisms then proceed to work with this summary of the robot's environment.

4. TWIG architecture

4.1. Overall structure

TWIG is divided into two parts. In the first part, the *extension finder*, TWIG parses an utterance and matches it to a fact observed in the environment. (Here and later, we use the word "fact" as a shorthand for "atomic sentence of predicate logic".) If a word is not understood but the sentence is close to matching a fact in the environment, the new word is inferred to refer to the object or relation in the environment that satisfies the meaning of the rest of the sentence. This part corresponds to the problem of finding the *extension* of the word, or its meaning under particular circumstances [8,13]. For example, if the first author of this paper were to say "I," the extension of "I" would be the individual named Kevin Gold. The extension finder can also find the two extensions connected by a transitive verb or preposition—for instance, if it heard "The pig foo the ball," it could find the particular objects in the environment that were being related by the word "foo," namely the pig and the ball.

The second part, the *definition tree generator*, takes as input the new words that the robot has heard and all the atomic sentences in the environment description that mention those words' extensions, and generates decision trees that represent their intensional meanings. Recall that the *intension* of a word is its general meaning, abstracted from any particular referent. Rather than building a decision tree for every individual word's meaning, the intension generator treats learning all the words for a particular part of speech as a single decision problem, with each word essentially defined by the decisions that would lead to it instead of a different word in the vocabulary.

The system assumes a predicate calculus representation for the robot's environment, and a semantics that defines words in terms of predicate calculus as well. This puts some burden on the robot designer in generating useful sensory predicates. However, the predicates are allowed to mention continuous values, for which the system later generates thresholds. This makes the system more useful in the real world than if it required all values to be discretely quantized.

The system assumes also that the robot has access to a speech recognition system that can interpret audio utterances as text. This prohibits the system from learning the semantics of words that it cannot even recognize—but the assumption here is that speech recognition systems will generally cover a much broader vocabulary than the robot has meanings for. Indeed, there is evidence to suggest that human infants tackle the speech segmentation problem before ever learning a semantics for the segmented words [40], so our system is in good company in treating segmentation and semantics as separatable issues.

We now turn to a more in-depth exposition of the two halves of the TWIG system.

4.2. TWIG part I: The extension finder

The extension finder functions very much like a standard discrete-clause grammar in parsing an utterance into logical form, with two major differences. First, the state of the world (in the form of a list of predicate calculus facts) is passed down the parse tree, and noun phrases as they are parsed are matched to known objects in the world. The symbols for these real-world objects then replace the words' intensional meanings during the parse. The fact that this replacement takes place before the parse is done means that the system can use some environmental context to narrow down the grammatical parse possibilities. The system can also backtrack on this match in case no parse succeeds with a particular word-to-object mapping.

If the parse fails, the system is allowed to retry the parse with a free variable. If the sentence contained a new word, this free variable resolves into whatever word extension is necessary to match a fact that the robot knows. This takes the form of a temporary term with a predicate named after the new word, and arguments named after its extensions—for example, `you(`*personR*`)` or `under(`*obj1, obj2*`)`. This temporary predicate allows the parse to succeed without the robot actually knowing what the word means; the system may not know what a "you" is, but it can infer that it's one of *those*.

Another reason the parse may fail is that all the words are known, but the fact that the sentence represents does not match anything that the robot knows. In this case, the same free variable can bind instead to the logical form of the whole sentence, which can then be added to the robot's knowledge base. In this way, understood sentences do not necessarily need to refer to things that the robot already knows, but can inform the robot of facts it cannot directly sense.

It is also possible that the sentence contains more than one new word, or contains a new word and also relates a fact that the robot does not know. In either case, the robot simply gives up on the sentence, and it has no further effect.

The extension finder is implemented in Prolog. The TWIG system adapts the following discrete-clause grammar from Pereira and Shieber [28]:

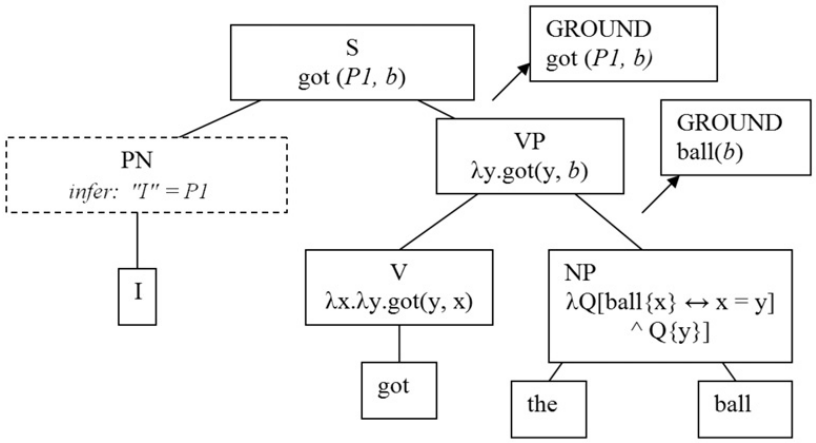

Fig. 3. Parsing a sentence with an undefined word, "I." The parse partially succeeds on the right, and the system finds that the whole sentence can be grounded if "I" refers to person *P1*, who has the ball. The missing definition, that "I" refers to the speaker, can only be learned over time.

```
s(S, W) --> np(VP^S, W), vp(VP, W).
np((E^S)^S, W) --> pn(E, W).
np(NP, W) --> det(N2^NP, W), n(N1).
vp(X^S, W) --> tv(X^IV), np(IV^S, W).
vp(S, W) --> [is], pp(S, W).
vp(IV, _W) --> iv(IV).
pp(X^S, W) --> p(X^PREP), np(PREP^S, W).
```

The abbreviations on phrase types are typical: *np* for "noun phrase," *iv* for "intransitive verb," and so on. *pn* covers both proper nouns and pronouns. W is a list of predicates indicating the state of the world.

A term in the form $X\hat{}\Phi$ is shorthand for the lambda expression $\lambda X.\Phi$, the notation for a function Φ with an argument X. In Montague's semantics (sometimes now simply called "formal semantics" [39]), the meanings of words and phrases could be expressed using lambda notation over logical expressions: the verb "has," for instance, could be expressed as $\lambda X.\lambda Y.possesses(X, Y)$, indicating that "has" refers to a function $possesses(X, Y)$ that takes two arguments, a possessor and possessed. In the Prolog language, these terms can be used inline in discrete-clause grammars, and the arguments of the functions are substituted as the parse provides them (see Fig. 3).

In the case of verbs and nouns, words at the lowest level are associated with their lambda calculus definitions, based on their parts of speech:

```
tv(Word, X^Y^pred([Word, X, Y])) :- member(vocab(tv,Word), W).
iv(Word, X^pred([Word, X])) :- member(vocab(iv,Word), W).
n(Word, X^pred([Word, X])) :- member(vocab(tv,Word), W).
p(Word, X^Y^pred([Word, X, Y])) :- member(vocab(p,Word), W).
```

Essentially, these statements relate the part of speech of a word to its logical semantics. For example, if a word *Word* is a transitive verb, then it should create a lambda calculus function of the form $\lambda X.\lambda Y.Word(X, Y)$, because it relates two different extensions, the subject and the object. (Transitive verbs that require an indirect object, such as "give," would need three arguments and their own grammatical category, but are not included in the grammar above.) A noun, on the other hand, contains only one argument, because it refers to just one thing: $\lambda X.Word(X)$. The Prolog statement `pred([P, ...])` represents the predicate $P(\ldots)$ in the robot's sensory representation; we shall see below that it is useful to treat the predicate P as a variable, and its arguments as a list.

Proper nouns, pronouns, and noun phrases beginning with "the" are immediately grounded in the robot's environment. In Prolog, this is expressed as follows:

```
det(the, W, (X^S1)^(X^S2)^S2) :- contains(W, S1).
pn(E, W) --> [PN], {pn(PN, W, E)}.
pn(PN, W, X) :- contains(W, pred([PN, X])).
```

The predicate `contains(W, X)` is true if the world W entails the fact X. A call to `contains` therefore implies a check against the robot's perceived reality, looking for an object that matches the description implied by a word. If an object to which the word or phrase applies is found, its symbol takes the place of the corresponding predicate. For instance, on parsing "the ball," the system searches the world W for a symbol X such that $ball(X)$. If $ball(b)$ is found in W, $X\hat{}ball(X)$ is replaced with b.

If the robot knows enough to understand the sentence S, the end result when the robot hears a sentence is that it is transformed into either the form `pred([P, X])` or the form `pred([P, X, Y])`, where X and Y are symbols that correspond directly to known objects and P is a sensory predicate. If the robot's world W contains this fact as well, nothing further happens. If W does not contain $P(X, Y)$, but there is some fact in W that contains P, the sentence is understood as new information, and is added to the knowledge base.

If the parse fails, the system is allowed to guess one word extension that it does not actually know. An unconstrained variable A is appended to the world W before passing it into the parser, and the parser solves for A. This effectively allows the robot to hypothesize a fact of the form `pred([Word, Object])`, where `Word` is the new word and `Object` is the object to which it refers.

Fig. 3 illustrates how this works. Suppose the robot hears the statement "I got the ball." It does not know who "I" is, but it sees girl a holding a ball b and girl e holding nothing. The parse fails the first time because the robot does not know the meaning of "I." It does, however, know that "got the ball" parses to $\lambda X.has(X, b)$. On retrying with the free variable, the robot finds that hypothesizing $I(a)$ allows it to match the sentence to $got(a, b)$, a fact it already knows. Thus, "I" is assumed to refer to a: the system has successfully inferred the extension.

In the case of new words with logical forms that take two arguments, such as transitive verbs and prepositions, the process works in the same way, only the system infers that the new word must be a relation between the two noun extensions found elsewhere in the parse. For instance, in parsing "The pig foo the ball," if the words "pig" and "ball" can be matched to specific objects p and b in the environment, the system will infer that the logical form of the whole sentence is `pred([foo, p, b])` even if it has not encountered the word "foo" before. This fact can be added as-is to the robot's knowledge base, but the fact is ultimately meaningless (ungrounded in sensory predicates) until the robot also learns the intension of "foo," which occurs in the next step.

In the process of parsing the sentence, the system must necessarily assign a part of speech to the new word as well. In the grammar described above, this choice is unambiguous. For more general grammars, the part of speech returned would be the first one found that satisfied the parse and produced a known fact.

The new word, its extension(s), and its part of speech are all passed to the next stage of the TWIG system for further processing.

4.3. TWIG part II: Definition tree generator

4.3.1. The interpretation of definition trees

Definition trees reconstruct the speaker's decision process in choosing a word for a referent or a relation. They are essentially decision trees, but they are built for use in the "reverse direction": the words are stored at the leaves, and a word's definition is given by the path from the root to the word's leaf. The interior nodes can be decisions about the referent's properties itself, or relations to other objects or people, particularly the speaker. When TWIG learns the meanings of new words, it does so by constructing definition trees that include the new words. The structures of these trees implicitly define the words.

Fig. 4 shows an example of a definition tree. Paths to the left after a decision indicate that the predicate is satisfied, while the right branch indicates that it is not. Each decision consists of attempting to satisfy a logical predicate with at most one threshold on a numerical argument to the predicate. We follow the additional convention that S always refers to the speaker, X always refers to a referent, Y is an optional second referent for the purpose of defining relations, and V is a numerical threshold. For example, one decision might be `dist(S,X,V) &` $V \leqslant 28.8$, indicating whether the distance between speaker S and referent X is less than 28.8 cm. (We will sometimes use the shorthand `dist(X,Y)<= V`, or omit mention of the threshold entirely if the attribute is boolean, assuming it to be $\geqslant 1$.) In choosing a word to describe X, the

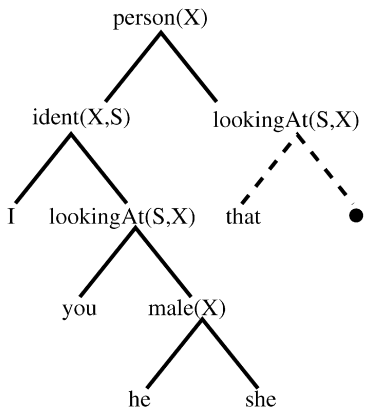

Fig. 4. An example of a definition tree. Paths to the left indicate that a predicate is satisfied. The predicates on the path to a word imply its definition. S refers to the speaker, and X refers to a referent. If the system reaches the dot when trying to decide what word to use for an entity, it decides that it has no word that applies.

system would decide whether speaker S and X satisfy this predicate and threshold. If the predicate is satisfied, the path on the left is followed; if they do not, the path on the right. This process continues until a leaf is reached, at which point the most common word at the leaf would be chosen.

The structure of a definition tree implies the logical definitions of the words it contains. Each word is defined as the conjunction of the predicates that lie on the path from the root to the leaf, with a predicate negated if the "no" path is followed. For example, the tree in Fig. 4 implies that the logical definition of "you" is

```
you(X) :- person(X) & ~ident(S,X) & lookingAt(S,X)
```

indicating that "you" is a person who is not the speaker, but whom the speaker is looking at. (We shall continue to use `ident` to refer to the identity relation.)

If the rightmost path of the tree is followed to its end, the system realizes that it does not know enough to describe the item in question; this is indicated by a dot in Fig. 4. The decision just before this final check is depicted in dotted lines to indicate that it is constructed in a slightly different manner from the other decisions. The system there is no longer contrasting words against other words, but choosing whether it knows enough about the referent to say anything at all.

The goal of the system is to construct such trees automatically from data consisting of utterances and the scenes that elicited them. We turn now to the algorithm for constructing these trees.

4.3.2. Constructing definition trees from data

Definition trees are constructed and updated using the output of the extension finding module. The construction method is essentially a variant on Quinlan's ID3 algorithm [32], but with the addition of the numerical thresholds of C4.5 [33], a final tree branch that catches cases in which no known word should apply, a choice of variables that depends on the extensions and part of speech, and some optimizations for online learning.

At each decision node, the available evidence is split into two groups based on the decision about the referent that is most informative to word choice, with one group satisfying the condition and the other not. This process then occurs recursively until no further decisions are statistically significant. The tree can be updated online, though the "batch" case shall be described first because it is simpler, and in some cases it must be called as a subroutine to the online version.

The output from the extension finder for a given utterance, and thus the input to the definition tree generator, is a 6-tuple $(W_i, T_i, X_i, Y_i, S_i, \Omega_i)$, where W_i is the new word, T_i is the inferred part of speech ("type"), X_i is the literal determined to be the referent, Y_i is an optional second referent (or null), S_i is the literal that was the speaker, and Ω_i is a list of predicates from the world that contain literals X_i, Y_i, or S_i as arguments. The second referent is non-null if the word was determined to refer to a relationship between two referents, instead of to a particular referent.

TWIG generates a separate decision tree for each part of speech. This is necessary because during language generation, the system should not attempt to use a noun as an intransitive verb, for instance, even though both have the same basic logical form and arguments. The rest of the exposition will assume we are dealing with the tree for a single part of speech.

Let t be an evidence 6-tuple; we now describe the function $D(t)$ that generates the set of possible decisions implied by the tuple. A *decision* here is a triple (P, V_0, σ), where P is a predicate with its arguments drawn exclusively from the set of variables $\{X, Y, S, V, \_\}$, V_0 is a threshold value, and σ is a sign indicating whether the direction of inequality for the threshold is $\geqslant$ or $\leqslant$. For each tuple, all instances of object X_i are replaced with the variable name X; all instances of object Y_i are replaced with variable name Y; instances of the speaker S_i are replaced with the variable S; and one numerical constant can be replaced with variable name V. All other arguments to the predicate are replaced with the anonymous variable "_". The threshold V_0 becomes the constant value that was replaced with the variable V, or 1 if there was no constant value. Then, two decisions are added to the set: one for $\sigma =$ "$\leqslant$" and one for $\sigma =$ "$\geqslant$". If a single literal plays more than one role, by being both speaker and the referent, then all possible decisions that can be generated by these substitution rules are added to $D(t)$. For example, if Bob was both the speaker and the sole referent of the new word, the term `inchesTall(bob, 60)` would generate four decisions: two decisions for whether the speaker (S) was at least or at most 60 inches tall, and two decisions for whether the referent (X) was at least or at most 60 inches tall. $D(t)$ consists of the union of all such sets generated by each predicate in Ω_i.

The algorithm must decide which of these decisions is most informative as to word choice. For each decision, a $2 \times |W|$ table is maintained, where $|W|$ is the number of unique words. This table maintains the counts of the instances in which the decision was satisfied for each word. Note that this requires attempting to satisfy each decision with the variables X, Y, S and V bound to their new values for each tuple.

From this table, one can easily calculate the information gain from splitting on the decision. Let W be the set of all words w_i, and let $w_i \in W_d$ if it was used under circumstances when decision d was satisfied, and $w_i \in W_{\neg d}$ otherwise. Then

$$Gain(d) = H(W) - \frac{|W_d|}{|W|} H(W_d) - \frac{|W_{\neg d}|}{|W|} H(W_{\neg d}) \tag{1}$$

where

$$H(W) = \sum_{i:\ w_i \in W} -\frac{|w_i|}{|W|} \log \frac{|w_i|}{|W|} \tag{2}$$

Readers may recognize $H(W)$ as the *entropy* of W, characterizing the average amount of information in a single word. $Gain(d)$ is the expected reduction in entropy on learning the truth or falsity of decision d. The decision with the most information gain is thus the fact about the referent that maximally reduces the "surprise" about the choice of word [20]. This is the criterion used by Quinlan for the original ID3 decision tree algorithm [32]. If two decisions are tied for informativeness, TWIG breaks the tie in favor of non-numerical predicates, thus penalizing the numerical predicates for complexity.

The maximally informative decision is added to the tree, so long as the decision and word choice are determined to be highly unlikely to be independent. A chi-square test can be computed using the same $2 \times |W|$ table to test for significance. TWIG uses Yates' continuity-corrected chi-square test [47]:

$$\chi^2_{Yates} = \sum_{i=1}^{N} \frac{(|O_i - E_i| - .5)^2}{E_i} \tag{3}$$

where the sum is over the cells of the table, O_i is the observed value in that cell, and E_i is the expected value in that cell under the assumption of independence. (Yates' corrected chi is a compromise between the normal chi square test, which is misleading for small sample sizes, and Fisher's exact test, which would take too long to compute for large numbers of decisions to be evaluated.)

Because the chi-square critical value depends on the number of degrees of freedom, which in turn depends on the number of unique words, a single critical value can't be used. Instead, the critical value τ is approximated using the following equations [24]:

$$\tau = \left(1 - A + \sqrt{A} P^{-1}(1-\alpha)\right)^3 \left(|W| - 1\right) \tag{4}$$

$$A = 2/\left(9\left(|W| - 1\right)\right) \tag{5}$$

Here $P^{-1}(\phi)$ is the inverse normal distribution, $|W|$ is the number of distinct words in the table, and α is the desired significance level. In the experiments to be described, $p < 0.001$ was arbitrarily set to be the threshold for significance.

With the decision added to the tree, the set of evidence tuples T is partitioned into two subsets—one subset consisting of the examples in which the decision was satisfied, and one in which the decision was not. These subsets are used at the left and right branches of the tree to generate subtrees, following the same procedure as outlined above. The process is then repeated recursively for each branch of the tree, until no more informative decisions can be added. The most common word among all the tuples at a leaf is chosen as the "correct" word for that leaf. (Irrelevant words at a leaf are often the result of sensor noise or speech recognition error.)

Once the basic tree has been recursively built, one more operation needs to be performed, in order to make the rightmost branch meaningful. Unmodified, the rightmost path in a basic definition tree always corresponds to a failure to prove anything about a referent, since the right path must be taken when a value is unknown. Though there exist some words that describe items about which nothing is known (e.g., "it," "thing"), they do not exist for all parts of speech; for example, there is no "default" transitive verb that implies nothing about its referents. Thus, some meaningful words may be assigned to this branch simply because there are no further distinctions to be made. Unmodified, this leaves the word mostly meaningless to the robot. Moreover, if the robot ever moved to a different environment or context, it would be constantly failing to prove anything, but then would mistakenly use its "default word" to describe every new situation.

For these reasons, the definition corresponding to the rightmost path is always augmented with the single non-negated decision that produces the highest information gain when contrasting the rightmost word with all other words in the tree. (This decision is represented in the decision tree diagrams with a pair of dotted lines.) The calculation uses all the evidence available at the root of the tree, but the $2 \times |W|$ table for each decision is collapsed into a 2×2 table, so that only the rightmost word vs. non-rightmost word decision is taken into account for informativeness.

4.3.3. Optimizations for online learning

The batch mode for tree generation is useful for reconstructing a tree from a datafile of evidence 6-tuples; most of the time, however, the tree is updated online in response to an utterance. If the $2 \times |W|$ tables for each decision (including decisions not chosen) are maintained at each node in the tree, a node update usually need only consist of updating these tables and added the few new decisions implied by the new tuple t. This update is performed first at the root, and then, if the root decision is unchanged, the new tuple is passed recursively down the tree to whichever branch it satisfies. The tree's structure only changes if, after updating the existing decision tables and adding the new decisions, the most informative decision at a node changes. In this case, the batch algorithm must be called for the entire subtree. While the worst-case running time for this online version is the same—theoretically, every update could change the most informative decision at the root, requiring the whole tree to be rebuilt—in practice, most updates do not change any decision nodes, and the update is fast because it only updates tables down a single path to a leaf.

With these optimizations in place, a Java-based implementation running on a 3.4 GHz Pentium 4 could typically update a tree with 100–200 evidence tuples in 1–2 seconds, with a worst case of about eight seconds. The implementation used no optimizations besides those described here, and probably could have been further sped up with a better optimized proof mechanism for checking the truth or falsity of decisions. (See also "Complexity," below.)

Note that despite these optimizations for online performance, *the order in which evidence is received does not matter* in determining the final state of the definition tree. The inputs could be shuffled, and the final tree would remain the same, because the structure of tree $n-1$ has no effect on the structure on tree n; it merely affects the time necessary to compute the nth tree. The table of evidence stored at the root contains all the information necessary to rebuild the tree from scratch. If new data is received that makes different decision at the root more informative than the current root decision, the entire tree is rebuilt using this data; if a new decision is more informative at a different interior node, the subtree stemming from that node is rebuilt. The online optimizations are efficient under the assumption that these are rare occurrences, but when the tree must be remade to better accommodate the data, it changes its structure to do so. TWIG makes use of its existing tree and data tables to avoid repeating calculations it has already made, but no decision is permanent. This also means that bad decisions introduced due to sensory error can be fixed or eliminated if the system is given more input.

4.3.4. Conversion to Prolog

After each definition tree update, the definition tree can be converted into Prolog, sent back to the extension generator, and asserted, so that the word meanings can be used for parsing immediately. For example, the meaning of "got" as implied by the definition tree in Fig. 5b becomes the following Prolog statement:

```
contains(W, pred([got, X, Y])) :-
member(W, pred([_P1, X])),
member(W, pred([_P2, Y])),
\+(contains(W, pred([ident, X, Y]))),
(contains(W, pred([dist, X, Y, V0])), V0 <= 38.0).
```

The "member" statements are necessary so that X and Y are instantiated before entering negated clauses. Prolog's negation-as-failure operator would otherwise dictate that `got(X,Y)` was false if `ident(X,Y)` held for *any* choice of X and Y in the environment – which is obviously not what is desired. If this definition had referred to the speaker, the clause `member(W, pred([says, S, _])` would perform the necessary binding.

A depth-first traversal of each definition tree can create an explicit list of valid words for each part of speech in the grammar. Explicit vocabulary lists prohibit the robot from using its internally defined predicates as words.

4.3.5. Complexity

We now turn to the analysis of complexity. The time to build the tree is the time to evaluate a single decision set, multiplied by the number of nodes in the tree. This includes leaves, since decisions are evaluated at the leaves but ultimately discarded. The maximum number of leaves in the tree is $|T|$, the number of evidence tuples; the maximum number of total nodes in the tree is therefore still $O(|T|)$. The number of decisions to be evaluated at each node is at most $O(|P|V)$, where P is the number of different predicates and V is the number of different values encountered for each predicate. (Despite the fact that there are several permutations of variables permissible for each predicate, that number is still bounded by a small constant.) Evaluating a decision requires in the worst case examining every fact in the environment, or $|E|$ steps. Thus, the complexity for the batch algorithm is $O(T^2V|P||E|)$. V typically will be linear in $|T||E|$, creating a running time of $O(T^3|P||E|^2)$; however, sampling a constant number of values for each predicate would bring the complexity down to $O(T^2|P||E|)$.

As mentioned previously, the worst case for the online version is a reorganization of the entire tree at each step, resulting in $O(T^2V|P||E|)$ steps for each new tuple. However, the more common case of an update that does not change the tree structure requires only $O(d|E|(|P|+|T|))$ operations to update, where d is the depth of the tree; the two added terms come from the time to update existing decisions at a node, $O(|E||P|)$, and the time to generate and evaluate new decisions from the new tuple, $O(|E||T|)$.

The linear dependence on the number of predicates and the size of the world means that the algorithm would scale well with the addition of a large factual database, containing information that the robot cannot directly perceive. However, if extended chains of inference are necessary to prove certain facts, the $|E|$ term would need to be replaced with the running time of whatever proof mechanism is used to evaluate the truth or falsity of decisions. Efforts at code optimization would be best directed at this proof procedure, since it is effectively the innermost loop of the tree update.

Practically speaking, most robots will probably have a small number of predicates, and the dominant terms will be the number of tuples observed so far and the size of the environment. In the online case, the time to update is usually linear in both of these quantities. Furthermore, as the number of examples increases, the chance of a major reorganization of the tree decreases, so that in the limit the updates will almost always be $O(d\|\Omega|(|P|+|T|))$, assuming value sampling. Thus, the algorithm should scale well in the long term to more words and more examples, as long as the number of numerical values is kept in check through sampling.

4.3.6. Starting trees

Clearly, if the algorithm is to understand all but one of the words in the sentence, it needs some word definitions to begin with. This implies that the algorithm needs some definition trees from the outset in order to use its word extension

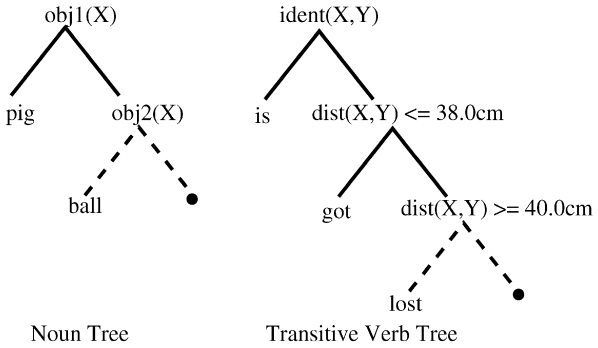

Fig. 5. The definition trees available to the system at the start of the experiment for (a) nouns and (b) transitive verbs.

finder. Unfortunately, merely specifying the structure of a starting tree is insufficient; the algorithm must have access to the data that generated a tree in order to update it.

The algorithm must therefore be initialized with some actual sensory data in which words are directly paired with their referents, circumventing the word extension finder. The starting vocabulary can be fairly minimal; our implementation began with five words, "ball," "pig," "is," "got," and "lost." The definition trees generated for these words (using 45 labeled examples) are shown in Fig. 5. Importantly, the data used to generate these trees must be generated under fairly similar conditions to the real online environment, or any consistent differences will be seen as highly informative. Ideally, the robot would be able to use pointing gestures or gaze direction to find these extensions without the extension finder, as this would presumably be most similar to how children learn these first examples in the absence of grammar; Yu and Ballard's gaze tracking word learner provided an excellent system for doing this, albeit not in a predicate logic framework [48]. In practice, it is not too difficult to manually provide words and referents for 45 different environment descriptions pulled from the robot's sensors.

An alternative, which TWIG used previously [15], is to have the starting definitions exist outside the definition tree structure, rendering their definitions unalterable and moot for the purposes of definition tree construction. This approach is less elegant, however, as the known definitions should ideally always inform the definitions of new words, and vice versa.

5. Experiment 1: Learning pronouns and prepositions

5.1. Methods

The TWIG system was initialized with the data necessary to produce the trees of Section 4.3.6 for nouns and transitive verbs; the pronoun and preposition trees began empty. For 200 utterances, the experimenters moved the stuffed pig and ball to different locations in the room, and then spoke one of the following utterances ([noun] should be understood to be "ball" or "pig"): ***This*** *is a [noun]*; ***That*** *is a [noun]*; ***I*** *got the [noun]*; ***You*** *got the [noun]*; ***He*** *got the [noun]*; *The [noun] is* ***above*** *the [noun]*; *The [noun] is* ***below*** *the [noun]*; *The [noun] is* ***near*** *the [noun].*

Words for which the system did not know definitions are in boldface; each utterance contained exactly one new word. Sentences that contained more than one new word, or no new words, were not included in the experiment, since these sentences could not affect the tree development—though speech recognition errors would sometimes result in such sentences being passed to TWIG.

Over the course of the experiment, the objects were moved arbitrarily about the room; positions included next to the robot, on the steps of a ladder, in the hands of one of the experimenters, on various tables situated about the room, and underneath those tables. The experimenters remained roughly 60 cm in front of the robot and 50–70 cm away from each other, and faced the appropriate individual (or robot) when saying "you" or "he." (This experiment was first reported in [15], but the decision tree algorithm has changed, and the comparison under "Evaluation" is new.)

5.2. Results

Many utterances were incorrectly recognized by Sphinx: at least 46%, based on a review of the system's transcripts. But because these false recognitions typically either included too many unknown words (e.g., "*He* is *near* the pig") or resulted in tautologies (e.g., "That is that"), the system usually made no inferences from them. A recognition error only affected tree development when it resulted in a sentence that contained exactly one unknown word.

Fig. 6 shows the state of the pronoun tree at the 27th, 40th, and final updates to the tree. The `person(X)` distinction remained the most informative attribute throughout the experiment, as it served to classify the two pronoun types into two broad categories. The proximal/distal distinction of "this" versus "that" was the next to be discovered by the system. The difference between "I," "you," and "he" remained unclear to the system for much of the experiment, because they relied on two unreliable systems: the sound localization system and the facing classifier, which had exhibited error rates of roughly 10% and 15%, respectively.

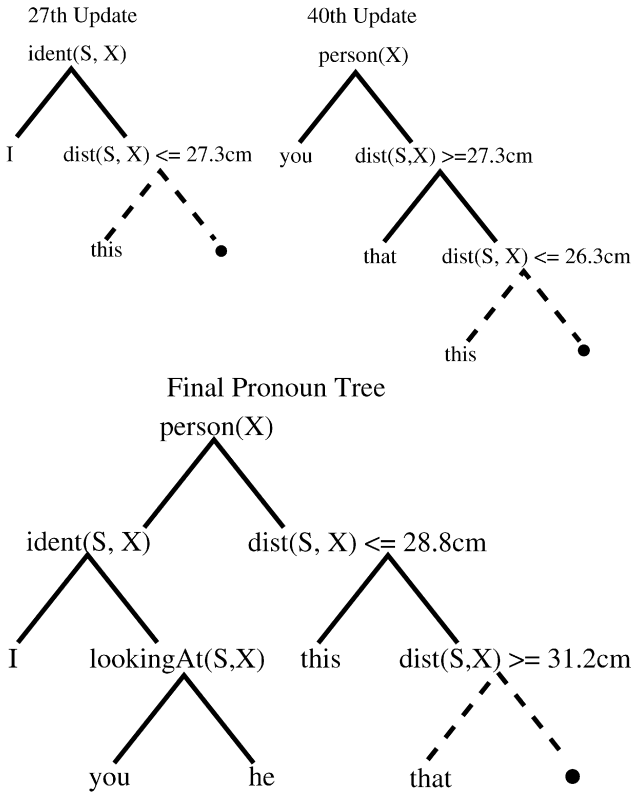

Fig. 6. The pronoun tree at the 27th, 40th, and final update. *S* refers to the speaker, and *X* to the word's referent; left branches indicate that the logical predicate is satisfied. "dist" refers to distance, and "ident" indicates the two terms are equal.

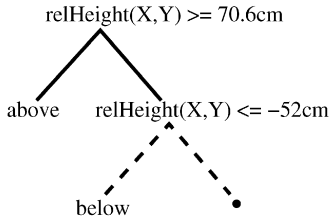

Fig. 7. The tree the system created to define prepositions.

The final definitions learned by the tree can be rendered into English as follows: "I" is the person that is the speaker. "You" is a person whom the speaker is looking at. "He" is a person who is not the speaker, and whom the speaker is not looking at. "This" is a non-person closer than 30 cm, and "that" is anything else.

The words "above," "below," and "near" were stored in a separate tree, shown in Fig. 7. Because there were no contrasting examples for "near," and in fact only four recognized utterances including the word, it did not receive its own leaf. However, the system did learn that "above" and "below" referred to the appropriate differences in relative height.

5.3. Analysis

The definitions implicit in these trees are for the most part correct, but were clearly limited by the robot's sensory capabilities and the nature of the data. For instance, "non-person that is closer than 28.8 cm" is clearly not a full definition of "this," which can also refer to farther away objects if they are large ("this is my house") or non-physical entities ("this idea of mine"). Under some circumstances, "you" may be used when the speaker is not looking at the addressee, and this subtlety is not captured here. Nevertheless, it is difficult to imagine better definitions being produced given the input available to the robot in this experiment.

The definitions are fairly subtle given the nature of the input, which contained no feedback from the speakers or "negative examples." The system learned that "he" should not refer to somebody the speaker is looking at, even though its only evidence was the fact that "you" was the preferred word for that case. It learned that "I," "you," and "he" must refer to people, and that "this" and "that" should not—again, constraints learned in the absence of negative feedback. (Admittedly, "this" can sometimes be used in introductions—e.g., "This is Kevin"—but in general it would be rude to ask someone, "Could this go get me a cup of coffee?") The system also demonstrated with "this" and "that" its ability to generate thresholds on numerical sensory data, in the absence of specific boolean predicates for proximity.

Table 2
TWIG's reaction to various speech recognition outcomes encountered during the experiment.

Sentence type	Example	TWIG reaction	Proportion
Understandable, accurate	I got the ball (true)	Accept	48%
Understandable, data mismatch	I got the ball (wrong speaker)	Accept	5%
Understandable, ambiguous reference	That is that	Accept	0.5%
More than one new word	He got that	Discard	28.5%
No extension produces valid fact	He got he	Discard	10.5%
Misheard as question	That is what	Discard	7.5%

The space of possible binary relations was not large, and so the reader may think it obvious that the system learned that height was chosen over distance and *lookingAt* as the definitions for "above" and "below." However, it is useful to note two things about this learning: first, that these words received fairly conservative thresholds due to the noisy nature of the Cricket input; and second, that "near" received no definition because every word in the tree implied some kind of nearness. (A pig on the other side of the room from the ball could not rightly be said to be "above" the ball.)

TWIG proved resilient against many speech recognition errors. Table 2 lists the different kinds of recognition errors encountered during the experiment, and TWIG's reactions to them. Because the space of possible utterances is much larger than the space of utterances that are meaningful in context, most speech recognition errors result in sentences that cannot be interpreted, and so they are discarded.

6. Experiment 2: Evaluation against variants

In addition to evaluating the TWIG system by the qualitative goodness of the definitions it produces, we examined the number and accuracy of the sentences it was able to produce about its environment, compared to similar systems.

6.1. Methods

The data generated by Experiment 1 was used to train four different word learning systems, each a modification of the core TWIG system. In one variant, TWIG's extension inference system was disabled, so that the words were not bound to any particular object or relation in the environment. The definition trees were created under the assumption that a decision was satisfied if *any* object in the environment satisfied the decision. For example, `dist(S,X) <= 30.0cm` was satisfied if the speaker was closer than 30 cm to any object. We call this strategy of associating everything in the environment with each word "associationist," using the terminology of [4].

As another variant, extension inference was allowed, but the system did not use definition trees. Instead, the system only used the single most statistically significant predicate to define each word, as measured by a chi-square test. This is the system that TWIG used prior to our invention of definition trees [17], and demonstrates the behavior of systems that attempt to learn each word as a separate learning problem, rather than treating all of word learning as a single decision problem.

Toggling whether extension inference was allowed and whether full definition trees were created resulted in four system variants, including TWIG itself. Each system was presented with four test environments, taken from actual robot sensory data. Each environment contained two people and the robot itself, as well as the ball and the pig. The ball and pig were each located either close to one of the people, close to the robot, or far away from any of them, and at varying heights. For each environment, each system produced all of the sentences that were true of that environment, according to the semantics it had learned. For evaluation, the systems were each provided with valid Prolog definitions for the words "is," "got," "ball," "lost," and "pig."

6.2. Results

Fig. 8 shows the results of this comparison. The associationist system that defined each word with a single predicate produced nonsensical definitions for every new word, but by chance managed to produce a fair number of correct sentences. The associationist tree-based system was more conservative and did not produce any more correct sentences than the tautologies implied by the definitions given to the system ("the pig is the pig"). The extension-finding system that defined each word by the single most informative predicate was much closer to the performance of TWIG, as it produced all the correct sentences that applied to each scene that TWIG did. However, because it could not represent conjunctions, its definitions of "this" and "that" omitted the requirement of being a person, while its definition of "he" was based purely on distance to the speaker. This resulted in sentences such as "that got the ball" (instead of "he got the ball") and "he is the ball."

TWIG's only errors in production were in the sentences "I is I" and "You is you" for each test environment, since it did not know the words "am" and "are." (Grammatically correct tautologies were counted as valid across all four conditions.) Other utterances it produced that it had never heard before included "I got this" and "That is below the ball."

Table 3 shows some sample utterances that each system produced about the situation shown in Fig. 9, with the robot next to the pig and speaking to the person on the left, who has the ball.

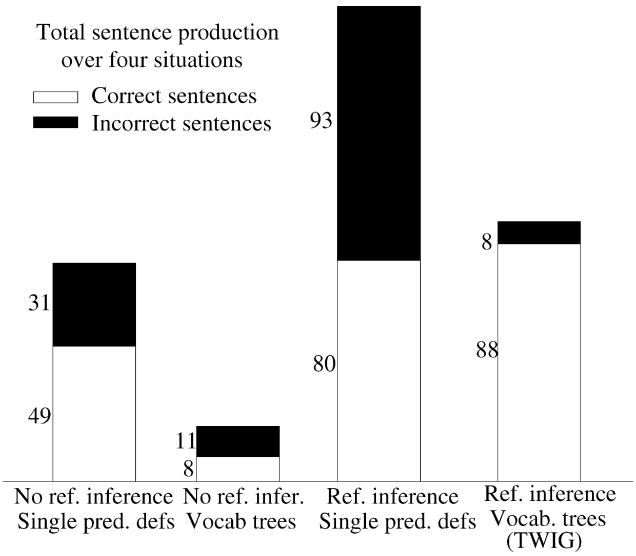

Fig. 8. Sentence production across four variants of TWIG. Disabling TWIG's ability to find word extensions generally results in incorrect definitions, while using single predicates instead of definition trees tends to result in less nuanced definitions, and hence, overproduction.

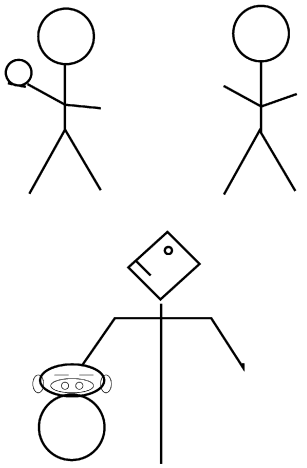

Fig. 9. A diagram of the test environment that produced the utterances in Table 3. The robot has the pig toy and is addressing the person on the left, who has the ball.

Table 3
Sample sentences produced by each system to describe the situation in Fig. 9. An asterisk indicates a statement counted as incorrect.

Associationist best predicate	Associationist word decisions	Extension-finding best predicate	TWIG (extension finding, word decisions)
*I lost the pig.	*I is above I.	I got the pig.	I got the pig.
*That got the ball.	*I is above the ball.	You got the ball.	You got the ball.
I lost the ball.	The ball is the ball.	*That lost he.	He lost the ball.
That is that.	*The pig is I.	*That got the ball.	You got that.
*That lost the pig.	*The pig is above the pig.	This is the pig.	This is the pig.

7. Experiment 3: Using new words as context to learn forms of "to be"

In Experiment 2, the two utterances which TWIG produced incorrectly across all four test situations were "I is I" and "you is you." There seemed to be no particular reason why the system could not learn the meanings of "am" and "are," adding them to its transitive verb tree; moreover, it seemed that the system might learn the correct conjugations of these words as well, with the structure of the tree dictating when each form of "to be" was appropriate based on relationship to the speaker.

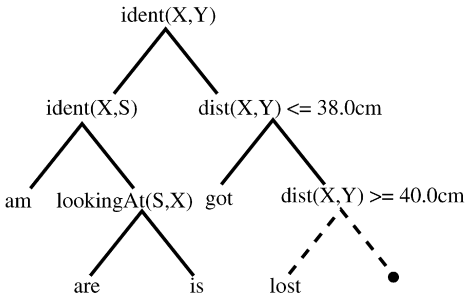

Fig. 10. The transitive verb tree after 20 examples of "I am I" and "you are you." Using the definitions for "I" and "you" learned in Experiment 1, the system infers that "am" and "are" refer to relations between an object and itself, and then introduces conjugation-like decisions as a means of distinguishing between them.

8. Methods

The definitions learned in Experiment 1 were loaded as Prolog word definitions. Instead of using the robot directly, the system cycled in simulation between the four scenes used for evaluation presented in Experiment 2, but with the dialog replaced with one participant saying either "I am I" or "you are you." The system was presented with 25 examples.

9. Results

As shown in Fig. 10, the system correctly learned that "am" and "are" referred to the identity relation. Moreover, the system learned the correct conjugation of these three forms of "to be" based on the same criteria that it used to distinguish between "I," "you," and "he": namely, that if the first referent was the speaker, use "am," and if the first referent was being spoken to, use "are"; otherwise, use "is." The tree reached this form after 20 examples.

It is also worth noting that the only sentence context available to learn these words were the words "I" and "you," and that using sentence context was essential to learn the meanings of these words, as it allowed the system to infer that in each case, the word referred to a relation between an entity and itself. However, neither the decisions used for conjugation nor the identity relation were imported from the pronoun tree, but were learned anew given the inferred reference.

10. Experiment 4: Learning context-sensitive definitions of colors

So far, we have discussed context in several senses. TWIG uses "sentence context," in which the meaning of the rest of the sentence informs the meaning of the word; "sensory context," in which the presence of sensory data allows better learning than the text alone; "relational context," in which relationships to other objects are taken into account when learning a word for a thing; "deictic context," a subset of relational contexts dealing specifically with the speaker; and "vocabulary context," in which a meaning of other words in the vocabulary can affect the meaning of a new word. Are these kinds of context sufficient to demonstrate the kind of context sensitivity desired by Roy and Reiter when they note that the meaning of "red" appears to differ depending on whether the phrase is "red wine, red hair, or red car" [36]?

The robot described in the previous experiments did not possess any means of object recognition, making this kind of distinction difficult to implement on the same robot as Experiment 1. Hence, this last experiment was performed with different sensory input entirely—namely, points of color sampled from Google image searches.

10.1. Methods

The following images were collected as samples from the first 20 hits of Google searches for the terms listed: 15 images of "red," 18 of "red hair," 17 of "red wine," 14 of "brown," 20 of "white hair," 19 of "white wine," 16 of "burgundy," 15 of "yellow," 15 of "brown hair," and 20 of "blonde hair." (The varying numbers were the result of excluding false positives, e.g., an image of Mr. Alan White.)

For each image, an arbitrary point in the relevant part of the image was sampled for its RGB values. This information was then converted into predicate form, and combined with a single predicate describing the kind of object depicted in the image. Thus, an image of red hair might receive the description `hair(obj), red(obj, 66), green(obj, 33), blue(obj, 16)`. The object classifications were hand labeled, and included *hair*, *wine*, and 37 other classifications.

These descriptions were then fed back to the decision-tree producing part of TWIG, bypassing the reference-finder, with the new color words bound in each case to the object variable. (This is equivalent to giving the full version of TWIG sentences of the form "The [noun] is [color]," assuming it possesses definitions matching the nouns to their predicates.)

10.2. Results

This data produced the tree shown in Fig. 11. The system did not produce separate definitions of red for wine and hair, despite the fact that the red of red wine might be better described as burgundy, and some red hair might otherwise be

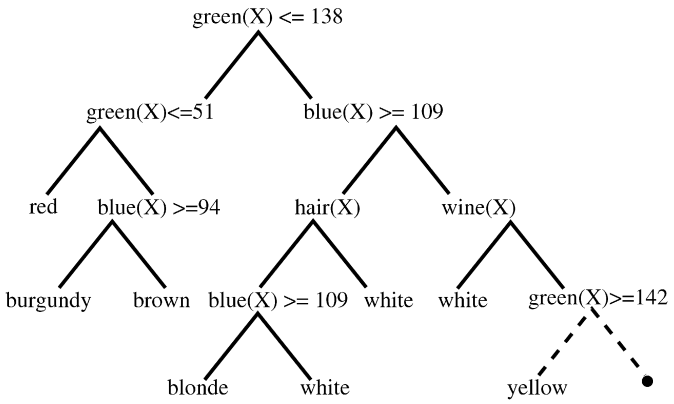

Fig. 11. An experiment on learning context-sensitive color words using the results of a Google image search. The tree includes three different branches for white: one for the shade of white hair, another for the shade of white wine, and a third for white in general.

called brown. This is presumably because the range shades called "red" spans and includes the red shades of red wine and red hair, and thus there was no need to distinguish these as special cases.

However, the system did produce three different definitions of white, based on the classification of the item in question. The system learned that some shades of white are better called "blonde" when hair is being described, and that something that appears yellow should be called "white" if it is wine. Thus, it appears that TWIG can indeed produce multiple definitions for words, with the word-to-definition mapping mediated by object category.

11. Discussion

To learn words in a complicated environment, a robotic word learning algorithm must have some way to determine what object or relation in the environment is being talked about. The mere presence or absence of an object in the environment is not enough, because many properties are continually present, but must be associated only with particular words. The malformed definitions created when we removed TWIG's ability to infer reference from context are a demonstration of this principle. Using those definitions resulted in the robot being unable to generate the rich variety of sentences it could form with extension-finding enabled.

One of the best ways to determine whether an object is relevant or not is by analyzing the rest of the sentence, which can strongly imply a particular object or relation as the target for a new word. For example, when TWIG hears "I got the ball," even when it does not know what "I" means in general, it can infer that the word refers to the person who has the ball. There are other methods of determining reference, such as following eye gaze direction, following pointing gestures, or using general perceptual salience, and these might be used to supplement the information available through sentence context. However, these other methods can be ambiguous, especially in the case of relations or deictic words. When the system hears "The ball is above the pig," it is easy to point to the ball and the pig, but hard to point to the "aboveness." Systems that can make sense of grammar have a clear advantage in the inference of new word meanings.

The addition of grammar is quite rare in sensor-grounded word learning, which reveals just how far this kind of research usually is from actual sentence production. Both in corpus-based text processing and sensor-based word learning, grammar is often absent, with sheer collocation in time used to find statistical associations [23,37,48]. Such a semantics does not lend itself to sentence production, because the learner does not learn how to combine its words to create new meanings. TWIG demonstrates that grammar can not only allow sentence production, but it also aids the comprehension of new words, particularly when words involving relations are involved. The tradeoff is that the grammar limits the system's ability to comprehend ungrammatical sentences, which are often common in human conversation. Nevertheless, it seems to make more sense to build a system with knowledge of grammar and hope to add robustness against ungrammaticality later, rather than give up on grammar from the start.

The reason grammar is so important is that it dictates not only how words must be combined, but how their meanings compose as well; it is the tip of the formal semantics iceberg. Formal semantics provides a much richer notion of meaning than the somewhat ad hoc semantics that have been used in robotic word learning systems to date. TWIG's separation of intension and extension, for instance, allows it to learn the meaning of "I," without being confused into thinking that "I" refers to a particular person all the time. Previous logic-based word learning systems would associate nouns with particular literal symbols [22,41], which is incorrect for all but proper nouns. On the other end of the spectrum, more sensor-based systems would produce interesting sensory prototypes for classes of objects, but then could not express relations between these objects or integrate more abstract properties into their definitions [37,48].

The primary difficulty with using a logic-based system in a real robot is that sensor data is rarely pre-packaged in the kind of neat formalism that formal semantics typically assumes. Real sensor data contains many numerical values and is often more low-level than the abstract predicates that word learning simulations commonly assume. TWIG gets around these difficulties to some extent by allowing definitions to include conjunctions of predicates and thresholds on numerical

values. Conjunctions allow high-level definitions to be built of lower-level predicates, while numerical thresholds bridge the gap between boolean-valued logic and continuous sensor readings.

Learning logical definitions for words can be difficult when the system is only given "positive" examples of each word, particularly when conjunction and negation are allowed in the definitions. Definitions that specify a value for every possible predicate are likely to be too specific: just because a word was learned on a Tuesday does not mean that `isTuesday(X)` should be informative. To address this issue, TWIG treats the word learning problem as one of making correct word *decisions*, with word meanings made more specific only when it is necessary to contrast one word with another. This approach appears to be similar to a heuristic young children use when learning new words, called the Principle of Contrast [9]. Though children appear to be partitioning the space of possible definitions, treating word learning as a single classification problem, this approach has not been previously used in robotic word learning, which has tended to treat each word as a separate recognition problem. Treating word learning as a problem of partitioning the semantic space allows faster learning with fewer examples and no explicit "inhibitory" or "negative" training. Definition trees have the additional advantages of being concise, quickly updated, easily understood, and specifying a word preference during generation when multiple words might apply.

Just as introducing grammar made the system less flexible in some ways, definition trees also carry a penalty: they assume that two different words of the same part of speech cannot refer to the same thing. Nevertheless, definition trees have the benefit of providing much more accurate definitions than our earlier system that did not make this assumption [17], which was essentially the "single best predicate" system tested in Section 6. Interestingly, human children appear to go through a phase of this assumption as well; in developmental psychology, it is known as the Principle of Mutual Exclusivity [26]. Perhaps further research into human language development will allow us to understand when, why, and how human infants abandon this assumption, so that we might incorporate their "fix" into TWIG. It is possible that these trees could still model which word should be used *first*, but that using a word should then free the system to travel down a second-best path in the tree; this approach may be useful for modeling anaphoric words such as "it." At any rate, creating a different tree for each part of speech alleviates some of the problem; for example, it allows "that" (pronoun) and "pig" (noun) to refer to the same object, as demonstrated in the experiment.

The rather artificial nature of the dialog in the experiment, with each participant repeatedly stating facts that are obvious from the environment, may cast some doubt on the generality of the word learning; real conversations often refer to things that are not in the immediate environment, and may include imperatives or questions in addition to declarative statements. There are three replies to this objection. First, the dialog in the experiment was restricted to achieve learning on a reasonable timescale, but the addition of language that the robot cannot understand would not hinder TWIG's learning, because TWIG only adds information to its trees if it can match the rest of the sentence to a fact it knows. Though waiting for such ideal statements from real dialog would be a slow process, the introduction of real speech should not have a negative impact on TWIG's learning, because most sentences would be ignored. Second, the robot can match statements to facts it knows that are not true of the immediate environment, if such facts are in its KB; obviously, more research needs to be done to ensure that the system could scale to a large KB, but note that TWIG in general will still only know a few facts about the *specific* extensions mentioned in the sentence and their relations to each other. Third, there is no reason in principle that the system could not be modified to learn from imperatives and questions, given some means of reasoning about such sentences; they are easily identified by their grammatical form, and their content might be inferred from the addressee's response.

TWIG is good at handling some kinds of ambiguity, but bad at others. When a reference is ambiguous, TWIG can make an incorrect extension inference, which could then lead to incorrect data to be used for determining meaning. It may be possible to simply avoid making inferences when a sentence contains ambiguous reference, but then again, referential ambiguity is common and this may not be a realistic solution. Perceptual uncertainty, on the other hand, can be essentially be treated as noise, which our experiments handled reasonably effectively, while ambiguity in meaning in the case of homophones could probably be determined by context and handled by creating separate definition tree branches for each meaning.

Some of the definitions implied by Fig. 6c may seem too simple, but they must be understood in the context of the robot's conceptual and sensory capacities. "I" has more connotations to a human than it does to Nico, but Nico's definition is sufficient to interpret nearby speakers' sentences or produce its own. The definitions of "this" and "that" are not as general as one might like, since they cannot capture reference to abstractions ("this idea") or take into account relativity of scale ("this great country"), but they do capture some interesting subtleties, such as the fact that "this" should not refer to the addressee even if he is close to the speaker. Moreover, the system can capture some of the context-sensitive nature of such words by creating new definition tree branches for particular kinds of referent, as we demonstrated in Section 5.3. Nevertheless, the use of absolute numerical thresholds to create dividing lines between word definitions is not a desirable property of TWIG, and a more "fuzzy" approach to semantic boundaries is an area for further exploration.

Given that our defense of the robot's definitions is built upon the robot's limited sensory capabilities, one might counter that the robot's world is therefore too simple to draw conclusions about learning in more complex environments. We would disagree that the robot's world is on the level of "Blocks World," as it includes several different noisy sensors producing continuous values and image processing routines that are genuinely useful (face detection, gaze direction). But even if Nico's world were taken to be very simple, the fact that systems that lack notions of reference or implicit contrast *fail* to learn correct pronoun meanings in even such a simple world, as demonstrated by our primary experiment, implies that such systems are highly unlikely to scale to a more complex environment, with even more features and objects. We do not claim

that our system is sufficient to learn the meanings of all possible words, but rather that extension inference and implicit contrast appear to be very useful, perhaps even necessary, for the unsupervised learning of certain word meanings.

The TWIG system's treatment of deictic, or speaker-relative terms, is somewhat novel within the world of formal semantics. Normally, a system such as Montague semantics handles deixis by assigning a sentence several "indices" within the space of possible speakers, times, and worlds; hence the term "indexical" for deictic pronouns [13]. This rather clunky approach stems from the fact that formal semantics is often used to analyze free-floating snippets of text, unconstrained by a physical environment and ambiguous in context. In a grounded sensory system, the solution is more straightforward: treat the speaker as another variable S to be grounded in the physical environment. In fact, most of the indices used to constrain indexicals could be restated as properties of the speaker: the speaker's time, place, topic of conversation, and so on. Moreover, the addition of the speaker variable S is useful even for words that aren't deictic pronouns. For example, it can allow the learner to take a speaker's attitude toward a referent into account: one can imagine a system in which `likes(S,X)` distinguishes the words "good" and "bad." The case of interjections is also interesting, because they have no extension at all; yet every language has words that convey `angry(S)`.

We have also experimented with introducing free variables beyond X, Y, and S in the sentence definitions. Such variables would potentially allow definition trees to learn the meaning of a phrase before learning its parts. For example, if the learner did not know the meaning of "have the ball," it might treat the whole thing as an intransitive verb meaning `got(X,A)&ball(A)`. Then, on hearing other phrases such as "have the pig" or "have the duckie", the system might notice the repetition of "have" at the beginning of each word in the subtree. It could then reorganize the tree to break off the leaves of `got(X,A)`, change "has" to a transitive verb meaning `got(X,Y)`, and reclassify its children as noun phrases. This is an idea which we are still developing, but it might allow the system to get around the one-word restriction and use whole sentences or phrases in the correct context without understanding their individual parts. Using our intensional definitions with variables that are not bound to specific objects in the environment might also allow "intensional reasoning" about general, rather than particular, members of a category.

It is also worthwhile to note in passing that this is the first system to learn the meaning of the word "I" from unsupervised observation, and then to use the word to refer to itself. In fact, the robot was not even provided with any training examples in which it was the speaker. We hope that the preceding discussion demonstrates that this accomplishment is not simply a parlor trick, but exists within a rich framework of language understanding.

The information-theoretic decisions used in definition trees may also provide a parsimonious explanation for several different heuristics observed among children. For example, young word learners cease to overextend particular words when they learn more apt ones, a heuristic known as the Principle of Contrast [9]. Similarly, young children reject perfectly good descriptions of objects if there are more appropriate words, a heuristic known as the Principle of Mutual Exclusivity [26]. The act of reasoning backwards about why a particular word was used is sometimes categorized as "theory of mind" [4]. Word choice based on maximal informativeness obeys Grice's Maxim of Quantity [18].

There are still many questions that remain to be explored with this system. How should words for superordinate categories such as "animal" be learned? How should the system choose the "next best word" if it has already used one, but wants to convey more information? How would the system work with raw audio, instead of text from a speech recognizer? What challenges await in learning concrete nouns from visual data? And can the "one unknown word" restriction be removed, so that the system can learn phrases before understanding their parts? These are all exciting avenues for future work.

Acknowledgements

Thanks to Justin Hart for helping with data collection. Support for this work was provided by a National Science Foundation CAREER award (# 0238334) and NSF award #0534610 (Quantitative Measures of Social Response in Autism). Some parts of the architecture used in this work was constructed under NSF grants #0205542 (ITR: A Framework for Rapid Development of Reliable Robotics Software) and #0209122 (ITR: Dance, a Programming Language for the Control of Humanoid Robots) and from the DARPA CALO/SRI project. This research was supported in part by a software grant from QNX Software Systems Ltd.

References

[1] J. Anglin, Vocabulary development: A morphological analysis, Monographs of the Society for Research in Child Development 58 (10) (1993) 1–166.
[2] D.R. Bailey, When push comes to shove: A computational model of the role of motor control in the acquisition of action verbs, Ph.D. thesis, Dept. of Computer Science, U.C. Berkeley, 1997.
[3] J. Bateman, Basic technology for multilingual theory and practice: The kpml development environment, in: Proceedings of the IJCAI Workshop in Multilingual Text Generation, 1995.
[4] P. Bloom, How Children Learn the Meanings of Words, MIT Press, Cambridge, MA, 2000.
[5] G. Bradski, A. Kaehler, V. Pisarevsky, Learning-based computer vision with Intel's open source computer vision library, Intel Technology Journal 9 (1), available online.
[6] R.W. Brown, Linguistic determinism and the part of speech, Journal of Abnormal and Social Psychology 55 (1) (1957) 1–5.
[7] S. Carey, The child as word-learner, in: M. Halle, J. Bresnan, G.A. Miller (Eds.), Linguistic Theory and Psychological Reality, MIT Press, Cambridge, MA, 1978.
[8] R. Carnap, Meaning and Necessity, University of Chicago Press, Chicago, 1947.

[9] E. Clark, The principle of contrast: A constraint on language acquisition, in: B. MacWhinney (Ed.), Mechanisms of Language Acquisition, Lawrence Erlbaum Assoc., Hillsdale, NJ, 1987, pp. 1–33.
[10] C. Crick, M. Doniec, B. Scassellati, Who is it? Inferring role and intent from agent motion, in: Proceedings of the 6th International Conference on Development and Learning, London, UK, 2007.
[11] C. Crick, M. Munz, B. Scassellati, Synchronization in social tasks: Robotic drumming, in: International Symposium on Robot and Human Interactive Communication, Hereford, England, 2006.
[12] C.G. de Marcken, Unsupervised language acquisition, Ph.D. thesis, MIT, 1996.
[13] D.R. Dowty, R.E. Wall, S. Peters, Introduction to Montague Semantics, D. Reidel, Boston, 1981.
[14] L. Fenson, P. Dale, J.S. Reznick, E. Bates, D. Thal, S. Pethick, Variability in early communicative development, Monographs of the Society for Research in Child Development 59 (5).
[15] K. Gold, M. Doniec, B. Scassellati, Learning grounded semantics with word trees: Prepositions and pronouns, in: Proceedings of the 6th International Conference on Development and Learning, London, UK, 2007.
[16] K. Gold, B. Scassellati, A Bayesian robot that distinguishes "self" from "other", in: Proceedings of the 29th Annual Meeting of the Cognitive Science Society (CogSci2007), Psychology Press, New York, 2007.
[17] K. Gold, B. Scassellati, A robot that uses existing vocabulary to infer non-visual word meanings from observation, in: Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07), AAAI Press, 2007.
[18] H.P. Grice, Logic and conversation, in: P. Cole, J. Morgan (Eds.), Syntax and Semantics, vol. 3: Speech Acts, Academic Press, New York, 1975, pp. 41–58.
[19] M.A.K. Halliday, An Introduction to Functional Grammar, Hodder Arnold, London, 1994.
[20] R.W. Hamming, Coding and Information Theory, second ed., Prentice-Hall, Englewood Cliffs, NJ, 1986.
[21] S.B. Heath, Ways With Words: Language, Life, and Work in Communities and Classrooms, Cambridge UP, New York, 1983.
[22] R.J. Kate, R.J. Mooney, Learning language semantics from ambiguous supervision, in: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-07), AAAI Press, Menlo Park, CA, 2007.
[23] T. Landauer, S. Dumais, A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge, Psychological Review 104 (2) (1997) 211–240.
[24] A.M. Law, D. Kelton, Simulation Modeling and Analysis, third ed., McGraw-Hill, New York, 2000.
[25] A. Lovett, B. Scassellati, Using a robot to reexamine looking time experiments, in: Proceedings of the 4th International Conference on Development and Learning, San Diego, CA, 2004.
[26] E.M. Markman, G.F. Wachtel, Children's use of mutual exclusivity to constrain the meanings of words, Cognitive Psychology 20 (1988) 121–157.
[27] W. O'Grady, How Children Learn Language, Cambridge UP, Cambridge, UK, 2005.
[28] F.C.N. Pereira, S.M. Shieber, Prolog and Natural-Language Analysis, CSLI/SRI International, Menlo Park, CA, 1987.
[29] S. Pinker, The Language Instinct, HarperCollins, New York, 1994.
[30] N.B. Priyantha, The Cricket indoor location system, Ph.D. thesis, Massachusetts Institute of Technology, 2005.
[31] W.V.O. Quine, Word and Object, MIT Press, 1960.
[32] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.
[33] J.R. Quinlan, Improved use of continuous attributes in c4.5, Journal of Artificial Intelligence Research 4 (1996) 77–90.
[34] T. Regier, The Human Semantic Potential: Spatial Language and Constrained Connectionism, MIT Press, Cambridge, MA, 1996.
[35] D. Roy, Semiotic schemas: A framework for grounding language in action and perception, Artificial Intelligence 167 (2005) 170–205.
[36] D. Roy, E. Reiter, Connecting language to the world, Artificial Intelligence 167 (1–2).
[37] D.K. Roy, A.P. Pentland, Learning words from sights and sounds: A computational model, Cognitive Science 26 (2002) 113–146.
[38] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, second ed., Prentice Hall, Upper Saddle River, NJ, 2003.
[39] J.I. Saeed, Semantics, second ed., Blackwell Publishing, Malden, MA, 2003.
[40] J.R. Saffran, R.N. Aslin, E.L. Newport, Statistical learning by 8-month-old infants, Science 274 (1996) 1926–1929.
[41] J.M. Siskind, Lexical acquisition in the presence of noise and homonymy, in: Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94), MIT Press, 1994.
[42] L. Steels, F. Kaplan, Aibo's first words: The social learning of language and meaning, Evolution of Communication 4 (1) (2001) 3–32.
[43] G. Sun, B. Scassellati, Reaching through learned forward model, in: Proceedings of the 2004 IEEE-RAS/RSJ International Conference on Humanoid Robots, Santa Monica, CA, 2004.
[44] P. Viola, M. Jones, Robust real-time face detection, International Journal of Computer Vision 57 (2) (2004) 137–154.
[45] T. Winograd, Computer program for understanding natural language, Ph.D. thesis, MIT, 1971.
[46] W.A. Woods, Semantics and quantification in natural language question answering, in: B.J. Grosz, K.S. Jones, B.L. Webber (Eds.), Readings in Natural Language Processing, Morgan Kaufmann, Los Altos, CA, 1978/1986, pp. 205–248.
[47] F. Yates, Contingency table involving small numbers and the chi square test, Journal of the Royal Statistical Society (Supplement) 1 (1934) 217–235.
[48] C. Yu, D.H. Ballard, A multimodal learning interface for grounding spoken language in sensory perceptions, ACM Transactions on Applied Perceptions 1 (1) (2004) 57–80.