Clarifying Commands with Information-Theoretic Human-Robot Dialog

Robin Deits[1]
Battelle Memorial Institute, Columbus, OH
Stefanie Tellex[1], Pratiksha Thaker, Dimitar Simeonov
MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA
Thomas Kollar
Carnegie Mellon University, Pittsburgh, PA
and
Nicholas Roy
MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA

Our goal is to improve the efficiency and effectiveness of natural language communication between humans and robots. Human language is frequently ambiguous, and a robot's limited sensing makes complete understanding of a statement even more difficult. To address these challenges, we describe an approach for enabling a robot to engage in clarifying dialog with a human partner, just as a human might do in a similar situation. Given an unconstrained command from a human operator, the robot asks one or more questions and receives natural language answers from the human. We apply an information-theoretic approach to choosing questions for the robot to ask. Specifically, we choose the type and subject of questions in order to maximize the reduction in Shannon entropy of the robot's mapping between language and entities in the world. Within the framework of the G^3 graphical model, we derive a method to estimate this entropy reduction, choose the optimal question to ask, and merge the information gained from the human operator's answer. We demonstrate that this improves the accuracy of command understanding over prior work while asking fewer questions as compared to baseline question-selection strategies.

Keywords: Human-robot interaction, natural language, dialog, information theory

1. Introduction

Our aim is to make robots that can naturally and flexibly interact with a human partner via natural language. An especially challenging aspect of natural language communication is the use of ambiguous reference expressions that do not map to a unique object in the external world. For

[1] The first two authors contributed equally to this paper.

instance, Figure 1 shows a robotic forklift in a real-world environment paired with instructions created by untrained users to manipulate one of the objects in the scene. These instructions contain ambiguous phrases such as "the pallet" which could refer equally well to multiple objects in the environment. Even if the human gives a command that would be unambiguous to another person, they might refer to aspects of the world that are not directly accessible to the robot's perceptions. For example, one of the commands in Figure 1 refers to "the metal crate." If a robot does not have access to perceptual features corresponding to the words "metal" or "crate," it cannot disambiguate which object is being referenced.

In this paper, we present an approach for enabling robots to avoid failures like these by asking a clarifying question, the same strategy that humans use when faced with ambiguous language. The robot first identifies the most ambiguous noun phrases in a command, and then asks a targeted question to try to reduce its uncertainty about which aspects of the external world correspond to the language. For example, when faced with a command such as "Move the pallet from the truck," in the situation shown in Figure 1, the robot can infer that the phrase "the pallet" is the most ambiguous, since there are two pallets in the scene and only one truck. It can then ask a question such as, "What do you mean by 'the pallet'?" The robot can use information from the answer to disambiguate which object is being referenced in order to infer better actions in response to the natural language command.

Previous approaches to robotic question-asking do not directly map between unconstrained natural language and perceptually-grounded aspects of the external world, and prior methods do not incorporate additional information from free-form natural language answers in order to disambiguate the command (Bauer et al., 2009; Doshi & Roy, 2008; Rosenthal, Veloso, & Dey, 2011). As a result, the robot cannot take advantage of its external world knowledge to determine the most ambiguous parts of an arbitrary natural language command and identify a targeted question to ask. Our approach, in contrast, takes an arbitrary natural language command as input. The robot derives a set of dialog actions to take based on that command. Different commands lead to a different set of dialog actions rather than relying on predefined dialog state-action space. The robot's strategy is adapted to the language and approach the person used in issuing the command.

In order to select an appropriate question to ask for a given command, our approach builds on the Generalized Grounding Graph (G^3) framework (Tellex, Kollar, Dickerson, Walter, Banerjee, A. G., Teller, S., & Roy, 2011; Tellex, Kollar, Dickerson, Walter, Banerjee, A., Teller, S., & Roy, 2011). The G^3 framework defines a probabilistic model that maps between parts of the language and *groundings* in the external world, which can be objects, places, paths, or events. The model factors according to the linguistic structure of the natural language input, enabling efficient training from a parallel corpus of language paired with corresponding groundings. In this paper, we use the G^3 framework to derive a metric based on entropy in order to estimate the uncertainty of the distribution of possible grounding values for the random variables in the model. The robot uses this metric to identify the most uncertain random variables in order to select a question to ask. Once the robot has asked a question, we show that it can exploit information from an answer produced by an untrained user by merging variables in the grounding graph based on linguistic coreference. By performing inference in the merged graph, the robot infers the best set of groundings corresponding to the command, the question, and the answer.

We evaluate the system using several different question-asking strategies: yes-or-no questions, targeted questions of the form, "What do you mean by X?" and reset questions, in which the robot requests that the human user rephrase the entire command. For yes-or-no questions, the system simulates the correct answer using ground-truth information; for other types of questions, we collected answers from human partners using crowdsourcing. We demonstrate that the system

(a)

```
Move the pallet from the truck.

Remove the pallet from the back of the truck.

Offload the metal crate from the truck.

Pick up the silver container from the truck bed
```

(b)

Figure 1. : Sample natural language commands collected from untrained users, commanding the forklift to pick up a pallet (a).

is able to incorporate information from the answer in order to more accurately ground concrete noun phrases in the language to objects in the external world. Furthermore, we show that our entropy-based metric for identifying uncertain variables to ask questions about significantly reduces the number of questions the robot needs to ask in order to resolve its uncertainty. This work expands on previous work presented in Simeonov, Tellex, Kollar, & Roy (2011) and Tellex et al. (2012) with the introduction and evaluation of two new types of questions (yes-or-no and reset, described in Section 3.1) and a new metric to select questions that will most effectively reduce the robot's uncertainty about its inferred sequence of actions (Metric 3 [Event Entropy], introduced in Section 3.1.2).

2. Background

We briefly review grounding graphs, which were introduced by Tellex, Kollar, Dickerson, Walter, Banerjee, A., Teller, S., & Roy (2011), giving special attention to the motivation for the use of a correspondence variable in the model definition. The correspondence variable, Φ makes it possible to efficiently train the model using local normalization at each factor but complicates the calculation of entropy described in Section 3.1.

In order for a robot to understand natural language, it must be able to map between words in the language and corresponding groundings in the external world. Each grounding g_i is a specific physical concept that is meant by some part of the language λ_j. Each grounding variable γ_i in the G^3 model takes a grounding g_i as its value. The goal of the system is to find the most probable groundings $g_1 \dots g_N$ for the grounding variables $\gamma_1 \dots \gamma_N$ given the language Λ and the robot's model of the environment m:

$$\underset{g_1 \dots g_N}{\operatorname{argmax}} \, p(\gamma_1 = g_1 \dots \gamma_N = g_N | \Lambda, m) \tag{1}$$

A random variable γ_i is created for each linguistic constituent in the parse tree of the natural language input Λ. The environment model m consists of the robot's location along with the locations and geometries of objects in the external world. A robot computes the environment model using sensor input. The computed model defines a space of possible values for the grounding variables, $\gamma_1 \dots \gamma_N$. Formally, each γ_i is a tuple, (r, t, p), where:

- r is a bounding prism. It is expressed as a set of points which define a polygon, $(x_1, y_1), \dots, (x_N, y_N)$, together with a height, z.
- t is a set of pre-defined textual tags, $\{tag_1, \dots, tag_M\}$, which are the output of perceptual classifiers.
- $p \in \mathbb{R}^{T \times 7}$ is a sequence of T points. Each point is a tuple (τ, x, y, z, r, p, y) representing the location and orientation of the object at time τ, represented as seconds since the epoch. Locations between two times are interpolated linearly.

To perform the inference in Equation 1, one standard approach is to factor it based on certain independence assumptions, and then use local models trained for each factor. Natural language has a well-known compositional, hierarchical argument structure (Jackendoff, 1985), and a promising approach is to exploit this structure in order to factor the model. However, if we define a directed, generative model over these variables, we must assume a possibly arbitrary order to the conditional γ_i factors. For example, a phrase such as "the tire pallet near the other skid," could be factored in either of the following ways:

$$p(\gamma_{\text{tires}}, \gamma_{\text{skid}} | \Lambda) = p(\gamma_{\text{skid}} | \gamma_{\text{tires}}, \Lambda) \times p(\gamma_{\text{tires}} | \Lambda) \tag{2}$$

$$p(\gamma_{\text{tires}}, \gamma_{\text{skid}} | \Lambda) = p(\gamma_{\text{tires}} | \gamma_{\text{skid}}, \Lambda) \times p(\gamma_{\text{skid}} | \Lambda) \tag{3}$$

Depending on the order of factorization, we will need different conditional probability tables that correspond to the meanings of words in the language. To resolve this issue, another approach is to use Bayes' Rule to estimate the $p(\Lambda | \gamma_1 \dots \gamma_N)$, but this distribution would require normalizing over all possible words in the language Λ. Another alternative is to use an undirected model, but this would require normalizing over all possible values of all γ_i variables in the model. This summation is intractable because there are an unbounded number of possible values for the γ_i variables: even if we assume a fixed set of object types, the number of possible object locations and configurations is infinite.

To address these problems, the G^3 framework introduces a correspondence vector Φ to capture the dependency between $\gamma_1 \dots \gamma_N$ and Λ. Each entry in $\phi_i \in \Phi$ corresponds to whether linguistic constituent $\lambda_i \in \Lambda$ corresponds to the groundings associated with that constituent. For example, the correspondence variable would be $True$ for the phrase "the tire pallet" and a grounding of an actual tire pallet, and $False$ if the grounding was a different object, such as a generator pallet. We assume that $\gamma_1 \dots \gamma_N$ are independent of Λ unless Φ is known. Introducing Φ enables factorization according to the structure of language with local normalization at each factor over a space of just the two possible values for ϕ_i. At inference time, these locally normalized factors can be simply

multiplied together without the need to compute a global normalization constant, as would be required for a Markov random field or conditional random field.

Using the correspondence variable, we can write:

$$\underset{g_1 \dots g_N}{\operatorname{argmax}} p(\gamma_1 = g_1 \dots \gamma_N = g_N | \Phi, \Lambda) \tag{4}$$

which is equivalent to maximizing the joint distribution of all groundings $\gamma_1 \dots \gamma_N$, Φ and Λ,

$$\underset{g_1 \dots g_N}{\operatorname{argmax}} p(\gamma_1 = g_1 \dots \gamma_N = g_N, \Phi, \Lambda). \tag{5}$$

We assume that Λ and $\gamma_1 \dots \gamma_N$ are independent when Φ is not known, as in the graphical model shown in Figure 2, yielding:

$$\underset{g_1 \dots g_N}{\operatorname{argmax}} p(\Phi | \Lambda, \gamma_1 = g_1 \dots \gamma_N = g_N) p(\Lambda) p(\gamma_1 = g_1 \dots \gamma_N = g_N) \tag{6}$$

This independence assumption is justified because if we do not know whether $\gamma_1 \dots \gamma_N$ correspond to Λ, then the language does not tell us anything about the groundings.

Finally, for simplicity, we assume that any object in the environment is equally likely to be referenced by the language, which amounts to a constant prior on $\gamma_1 \dots \gamma_N$.[2] We ignore $p(\Lambda)$ since it does not depend on $\gamma_1 \dots \gamma_N$, leading to:

$$\underset{g_1 \dots g_N}{\operatorname{argmax}} p(\Phi | \Lambda, \gamma_1 = g_1 \dots \gamma_N = g_N) \tag{7}$$

We factor the model according to the hierarchical, compositional linguistic structure of the command:

$$p(\Phi | \Lambda, \gamma_1 \dots \gamma_N) = \prod_i p(\phi_i | \lambda_i, \gamma_{i_1} \dots \gamma_{i_k}) \tag{8}$$

The specific random variables and dependencies are automatically extracted from the parse tree and constituent structure of the natural language command; the details of this factorization are formally described by Tellex, Kollar, Dickerson, Walter, Banerjee, A., Teller, S., & Roy (2011). Parses can be extracted automatically, for example with the Stanford Parser (Marneffe, MacCartney, & Manning, 2006) or annotated using ground-truth parses, as we do for the evaluation in this paper. We call the resulting graphical model the *grounding graph* for the natural language command. Figure 2 shows the parse tree and graphical model generated for the command, "Pick up the pallet." The random variable ϕ_2 is associated with the constituent "the pallet" and the grounding variable γ_2. The random variable ϕ_1 is associated with the entire phrase, "Pick up the pallet" and depends on both grounding variables: γ_1, which is the action that the robot takes, and its argument, γ_2, which is the object being manipulated. The λ_i variables correspond to the text associated with each constituent in the parse tree.

We assume that each factor takes a log-linear form with feature functions f_j and feature weights θ_j.

$$p(\Phi | \Lambda, \gamma_1 \dots \gamma_N) = \prod_i \frac{1}{Z} \exp(\sum_j \theta_j f_j(\phi_i, \lambda_i, \gamma_{i_1} \dots \gamma_{i_k})) \tag{9}$$

[2] In the future, we plan to incorporate models of attention and salience into this prior.

This function is convex and can be optimized with gradient-based methods (McCallum, 2002). Training data consists of a set of natural language commands together with positive and negative examples of groundings for each constituent in the command.

Features correspond to the degree to which the $\gamma_1 \dots \gamma_N$ correctly ground λ_i. These features define a *perceptual representation* in terms of mapping between the grounding and words in the language. For example, for a prepositional relation such as "on," a natural feature is whether the grounding corresponding to the head noun phrase is supported by the grounding corresponding to the argument noun phrases. However, the feature '$supports(\gamma_i, \gamma_j)$' alone is not enough to enable the model to learn that "on" corresponds to '$supports(\gamma_i, \gamma_j)$'. Instead, we need a feature that also takes into account the word "on" so that,

$$supports(\gamma_i, \gamma_j) \wedge (\text{“on”} \in \lambda_i) \tag{10}$$

Thus features consist of the Cartesian product of perceptual features such as $supports$ crossed with the presence of words in the linguistic constituent associated with the corresponding factor in the grounding graph.

This system follows natural language commands by optimizing the objective in Equation 7. It carries out approximate inference by performing beam search over $\gamma_1 \dots \gamma_N$. It searches over possible bindings for these variables in the space of values defined in the environment model M. It then computes the probability of each assignment using Equation 7; the result is the maximum probability assignment of values to all the variables $\gamma_1 \dots \gamma_N$. Although we are using $p(\Phi|\Lambda, \gamma_1 \dots \gamma_N)$ as the objective function, Φ is fixed, and the $\gamma_1 \dots \gamma_N$ are unknown. Given our independence assumptions, this approach is valid because $p(\Phi|\Lambda, \gamma_1 \dots \gamma_N)$ corresponds to the joint distribution over all the variables given in Equation 5. We discretize the space of possible groundings to make this search problem tractable. If no correct grounding exists in the space of possible values, then the system will not be able to find the correct value; in this case it will return the best value that it found.

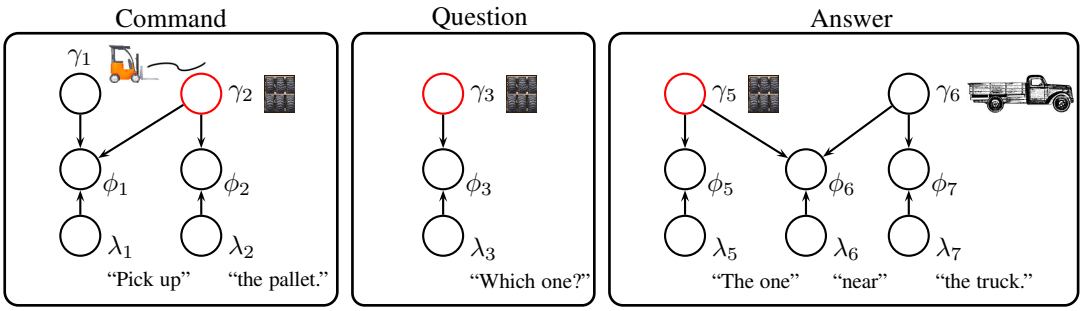
3. Technical Approach

When faced with a command, the system parses the language into the corresponding grounding graphs and performs inference to find the most likely set of values for the grounding variables $\gamma_1 \dots \gamma_N$. The results described in this paper use ground-truth syntax parses, but automatic parsing strategies are also possible.[3] Next, the system identifies the best question to ask using an entropy-based metric and asks it, as described in Section 3.1. We describe and analyze three such metrics for selecting questions in Sections 3.1.1 and 3.1.2. After asking the chosen question and receiving an answer from a human partner, the robot merges grounding graphs that correspond to the original command, question, and answer into a single graphical model. Finally, the system performs inference in the merged graph to find a new set of groundings that incorporates information from the answer as well as information from the original command. Figure 3 shows the dataflow in the system.

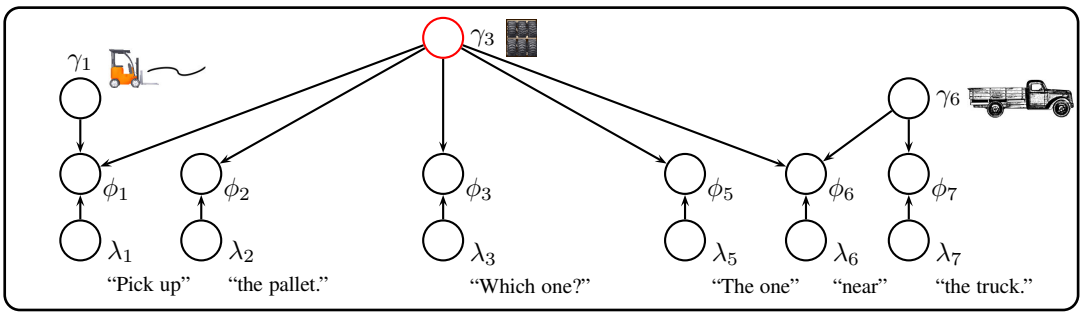
3.1 Generating Questions

In this paper we consider three general categories of questions: yes-or-no, targeted, and reset. A yes-or-no question asks the user for confirmation of the correspondence between a particular

[3] In our previous work, we showed that the Stanford Parser (Marneffe, MacCartney, & Manning, 2006) could be used to parse commands at the cost of a roughly 10% penalty in command understanding accuracy. However answers to questions are often incomplete sentences that do not match well with the training set used by the automatic parser. We used ground-truth parses to focus the evaluation on the semantics and question-asking parts of the system rather than parsing accuracy which is not a focus of our research.



(a) Unmerged grounding graphs for three dialog acts. The noun phrases "the pallet," "one" and "the one near the truck" refer to the same grounding in the external world but initially have separate variables in the grounding graphs.



(b) The grounding graph after merging γ_2, γ_3 and γ_5 based on linguistic coreference.

Figure 2. : Grounding graphs for a three-turn dialog, before and after merging based on coreference. The robot merges the three shaded variables.

grounding variable γ_j and a grounding in the world. The system does so by asking about the linguistic constituent to which the grounding variable is connected in the G^3 framework, such as "Do the words 'the box' refer to this generator pallet?" A targeted question prompts the user for an open-ended description of a single grounding variable γ_j, such as "What do the words 'the box' refer to?" Finally, a reset question simply asks the user to restate the command in different words: "I didn't understand. Can you please rephrase the command?"

Given a question type, the robot's aim is to choose a question from which the answer will provide it with the most information from the human user. (We will discuss the challenge of choosing a type of question to ask in Section 3.1.2.) The robot must pick a specific question from a space of possible questions defined by the natural language command and objects in the environment. The robot can ask a yes-or-no question about any pair of grounding variable γ_j and candidate value, g. Likewise, it can ask a targeted question about any grounding variable γ_j. There is only one possible reset question to ask.

3.1.1 Selecting a Grounding Variable One intuitive estimate for the uncertainty of a grounding variable γ_j is to look at the probability of the correspondence variable ϕ_k for each factor it participates in. By this estimate, the most uncertain variable γ_j can be found as follows:

$$\underset{j}{\operatorname{argmin}} \prod_{k \in factors(\gamma_j)} p(\phi_k = T | \gamma_1 = g_1 \dots \gamma_N = g_N, \Lambda) \tag{11}$$

where $g_1 \dots g_N$ are the groundings generated by the inference in Equation 7.

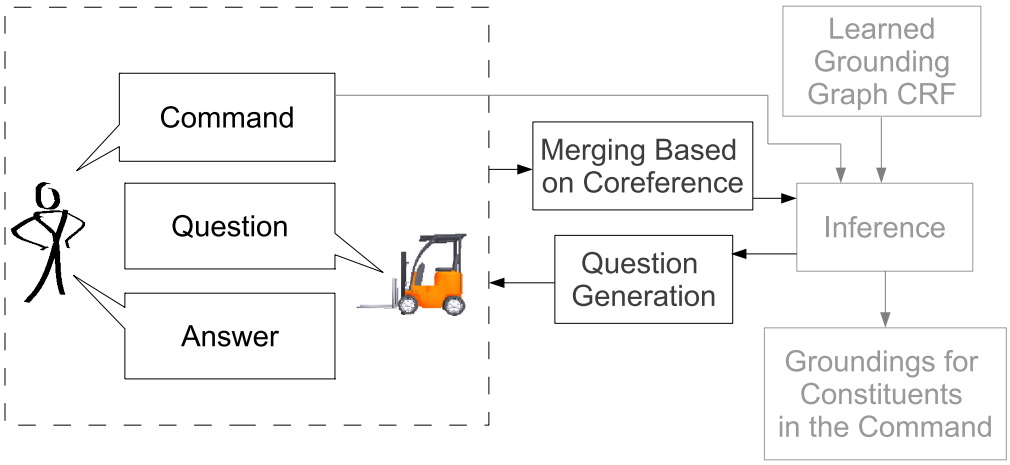


Figure 3. : System diagram. Grayed-out blocks show components developed in previous work and are therefore not discussed in detail in this paper; black blocks show the question-asking feedback system new to this paper.

Here the system asks questions about variables for which it was unable to find a high-probability grounding for variable γ_j. For targeted questions, choosing grounding variable γ_j is sufficient to generate a question. For yes-or-no questions, the system must also choose an object to ask about; here we select the grounding value g with the highest probability from the space of possible values:

$$\underset{g}{\operatorname{argmax}} \underset{j}{\operatorname{argmin}} \prod_{k \in factors(\gamma_j)} p(\phi_k = T | \gamma_1 \dots \gamma_{j-1}, \gamma_j = g, \gamma_{j+1} \dots \gamma_N, \Lambda) \tag{12}$$

We refer to this approach as **Metric 1**. However, this metric will not perform well if there are several objects in the external environment that could correspond equally well to the words in the language. As an example, a vague expression such as "the pallet" would have high confidence for any pallet that was grounded to it. But if there were many pallets in the environment, the robot might be very uncertain about which one was meant.

A more principled approach is to formally derive an expression for the entropy of the distribution over each grounding variable, and then ask a question about the variable with maximum entropy. Entropy directly quantifies the robot's uncertainty; searching for questions that minimize entropy directly targets the parts of the command where the robot is most unsure. We begin by defining a family of marginal distributions for each grounding variable γ_j conditioned on Φ and Λ:

$$p(\gamma_j | \Phi, \Lambda) \tag{13}$$

To find the most uncertain grounding variable γ_j, we find the distribution in this family with the highest entropy:

$$\underset{j}{\operatorname{argmax}} \, H_{p(\gamma_j | \Phi, \Lambda)}(\gamma_j) \tag{14}$$

The system can collect more information from the human partner in order to disambiguate the command by asking a question about the most uncertain variable. For example, if a command like "bring the pallet on the truck to receiving" were issued in a context with two trucks and one pallet, the entropy would be higher for the phrase "the truck"and the system could ask a question such as "What do you mean by 'the truck'?" On the other hand, if there were two pallets and one truck, the entropy would be higher for the phrase "the pallet" and the system would ask a question such as "What do you mean by 'the pallet'?"

We can expand the entropy function as:

$$H_{p(\gamma_j|\Phi,\Lambda)}(\gamma_j) = -\sum_{\gamma_j} p(\gamma_j|\Phi,\Lambda)\log p(\gamma_j|\Phi,\Lambda) \tag{15}$$

Unfortunately, $p(\gamma_j|\Phi,\Lambda)$ cannot be directly computed in the G^3 framework because during inference, the system maximizes $p(\Phi|\Lambda,\gamma_1\dots\gamma_N)$, with respect to the unknown grounding variables $\gamma_1\dots\gamma_N$. Instead, we rewrite it as a marginalization over the joint:

$$p(\gamma_j|\Phi,\Lambda) = \sum_{\gamma_1\cdots\gamma_{j-1},\gamma_{j+1}\cdots\gamma_N} p(\gamma_1\dots\gamma_N|\Phi,\Lambda) \tag{16}$$

We use Bayes' rule to rewrite the equation as:

$$p(\gamma_j|\Phi,\Lambda) = \sum_{\gamma_1\cdots\gamma_{j-1},\gamma_{j+1}\cdots\gamma_N} \frac{p(\Phi|\gamma_1\dots\gamma_N,\Lambda)p(\gamma_1\dots\gamma_N|\Lambda)}{p(\Phi|\Lambda)} \tag{17}$$

Next, we assume that $\gamma_1\dots\gamma_N$ are independent of Λ when we do not know Φ, as we did in Equation 6, yielding:

$$p(\gamma_j|\Phi,\Lambda) = \sum_{\gamma_1\cdots\gamma_{j-1},\gamma_{j+1}\cdots\gamma_N} \frac{p(\Phi|\gamma_1\dots\gamma_N,\Lambda)p(\gamma_1\dots\gamma_N)}{p(\Phi|\Lambda)} \tag{18}$$

Finally, we assume a constant prior $p(\gamma_1\dots\gamma_N) = C$ and define a constant $K = C/p(\Phi|\Lambda)$:

$$p(\gamma_j|\Phi,\Lambda) = K\sum_{\gamma_1\cdots\gamma_{j-1},\gamma_{j+1}\cdots\gamma_N} p(\Phi|\gamma_1\dots\gamma_N,\Lambda) \tag{19}$$

Precisely computing this quantity requires summing over all possible values of all grounding variables except for γ_j for a particular environment and is intractable. Instead, we efficiently approximate this summation based on the results of the inference process. After running inference, the system saves all M sets of values that it considered for grounding variables in the model. Each sample s_m consists of bindings for grounding variable γ_j in the grounding graph, and we denote the value of variable γ_j in s_m as $s_m[\gamma_j]$. When performing inference using beam search, we set the beam width to 10. Our results demonstrate that this beam width creates sufficiently diverse samples to provide an entropy estimate which enables effective question selection. We approximate Equation 19 with:

$$\hat{p}(\gamma_j = g|\Phi,\Lambda) = \frac{c(\gamma_j = g,\Phi,\Lambda)}{\sum_x c(\gamma_j = x,\Phi,\Lambda)} \tag{20}$$

where c is

$$c(\gamma_j = g, \Phi, \Lambda) = K \sum_{\{\, s_m | s_m[\gamma_j] = g \,\}} p(\Phi | s_m[\gamma_1] \dots s_m[\gamma_N], \Lambda) \times p(s_m[\gamma_1] \dots s_m[\gamma_N]) \tag{21}$$

We can then substitute this into Equation 15 to obtain an estimate for the entropy. We refer to this approximation as **Metric 2**.

For a yes-or-no question, we must specify both the grounding γ_j and the entity g that are being asked about. Our approach is to find:

$$\arg\max_{\gamma_j, g} H_{p(\xi(\gamma_j, g)|\Phi,\Lambda)}(\xi(\gamma_j, g)) \tag{22}$$

where $\xi(\gamma, g)$ is defined to be a binary random variable with value 1 if and only if $\gamma = g$. We can calculate this entropy as follows:

$$\begin{aligned} H_{p(\xi(\gamma_j, g)|\Phi,\Lambda)}(\xi(\gamma_j, g)) &= -p(\gamma_j = g|\Phi, \Lambda) \log p(\gamma_j = g|\Phi, \Lambda) \\ &\quad - p(\gamma_j \neq g|\Phi, \Lambda) \log p(\gamma_j \neq g|\Phi, \Lambda) \qquad (23) \\ &= -p(\gamma_j = g|\Phi, \Lambda) \log p(\gamma_j = g|\Phi, \Lambda) \\ &\quad - (1 - p(\gamma_j = g|\Phi, \Lambda)) \log (1 - p(\gamma_j = g|\Phi, \Lambda)) \qquad (24) \end{aligned}$$

We compute Equation 24 using the approximation in Equation 20.

3.1.2 Estimating Event Entropy The metrics presented so far attempt to identify the most uncertain grounding variable about which to ask. However, if the ultimate goal of the system is to produce a correct action for a natural language command, then a more natural metric is to consider the entropy of the possible actions. Consider a command such as "Pick up the generator pallet near the stack of boxes." If "generator pallet" can be uniquely and correctly identified by the system, then any ambiguity in "the stack of boxes" is largely irrelevant: since the robot knows which pallet is being indicated, it will perform the correct action. Thus, we propose a more general metric to measure the quality of a potential question: the expected entropy over actions the robot could take in response to a natural language command. We refer to the random variable corresponding to the overall, top-level action as γ_e and search for a question, q, which minimizes entropy over this specific variable on expectation over all possible answers, a:

$$\operatorname*{argmin}_q E_a(H_{p(\gamma_e|\Phi,\Lambda,q,a)}(\gamma_e)) \tag{25}$$

This quantity has the additional advantage of providing a common metric to compare the effectiveness of asking questions of different types, and more generally, of other non-linguistic information gathering actions. However, Equation 25 is not necessarily practical to compute. Since open-ended questions by definition have a limitless space of possible answers, calculating the expectation in Equation 25 is generally intractable. Approximations require some kind of model for the types of answers expected from a particular human partner.

However, for yes-or-no questions, the limited number of possible pairings of grounding variables in the graphical model and objects in the action space of the robot means that the space of questions and answers can be fully explored. To compute the final event entropy for yes-or-no questions, for

each grounding variable γ_j and for each grounding g to which that variable may bind, we estimate the probability that $\gamma_j = g$ using Equation 20. Thus, we can estimate that if a yes-or-no question were asked about the correspondence between γ_j and g, the probability of receiving a 'yes' answer should be $\hat{p}(\gamma_j = g|\Phi, \Lambda)$, and the probability of a 'no' should be $1 - \hat{p}(\gamma_j = g|\Phi, \Lambda)$.

We can then estimate the event entropy $H(\gamma_e)$ in the event of either answer:

$$\text{'Yes':} \quad H_{\text{yes}} = H_{p(\gamma_e|\gamma_j=g,\Phi,\Lambda)}(\gamma_e) \quad \approx -\sum_{\gamma_e} \hat{p}(\gamma_e|\gamma_j = g, \Phi, \Lambda) \log \hat{p}(\gamma_e|\gamma_j = g, \Phi, \Lambda) \quad (26)$$

$$\text{'No':} \quad H_{\text{no}} = H_{p(\gamma_e|\gamma_j\neq g,\Phi,\Lambda)}(\gamma_e) \quad \approx -\sum_{\gamma_e} \hat{p}(\gamma_e|\gamma_j \neq g, \Phi, \Lambda) \log \hat{p}(\gamma_e|\gamma_j \neq g, \Phi, \Lambda) \quad (27)$$

where

$$\hat{p}(\gamma_e = a|\gamma_j = g, \Phi, \Lambda) = \frac{c(\gamma_e = a, \gamma_j = g, \Phi, \Lambda)}{\sum_x c(\gamma_e = x, \gamma_j = g, \Phi, \Lambda)} \quad (28)$$

and

$$c(\gamma_e = e, \gamma_j = g, \Phi, \Lambda) = K \sum_{\{\, s_m | s_m[\gamma_e=a], s_m[\gamma_j]=g \,\}} p(\Phi|s_m[\gamma_1] \dots s_m[\gamma_N], \Lambda) \times p(s_m[\gamma_1] \dots s_m[\gamma_N]) \quad (29)$$

likewise for $\gamma_j \neq g$.

The expected final event entropy for a given choice of γ_j and g is

$$\begin{aligned} E(H_{p(\gamma_e|\Phi',\Lambda')}(\gamma_e)) = \hat{p}(\gamma_j = g|\Phi, \Lambda)\, H_{\text{yes}} \\ + (1 - \hat{p}(\gamma_j = g|\Phi, \Lambda))\, H_{\text{no}} \end{aligned} \quad (30)$$

which we can calculate for all possible pairings of γ_j and g. We refer to this method of question selection as **Metric 3**.

3.1.3 Generating Question Text After selecting a grounding variable γ_j to ask about, the robot asks a question using a template-based algorithm. The structure of the G^3 model allows the system to identify the language parts λ_j in the original command to which the grounding variable corresponds. For a yes-or-no question, the robot assumes access to a deictic gesture that uniquely identifies an object and generates a question of the form, "Do the words 'X' refer to this one?" For a targeted question, the robot generates a question of the form, "What do the words 'X' refer to?" Once a question has been generated, the system asks it and collects an answer from the human partner. While we assume answers to yes-or-no questions are either yes or no, answers to targeted and reset questions could take many forms. For example, Figure 4 shows commands and questions generated using the template-based algorithm, along with corresponding answers collected from untrained users.

3.2 Understanding Answers to Questions

Once a question has been chosen and an answer obtained, the system incorporates information from the answer into its inference process. The system begins by computing separate grounding graphs for the command, the question, and the answer according to the parse structure of the language. Next, variables in separate grounding graphs are merged based on linguistic coreference. Finally, the system performs inference in the merged graph to incorporate information from the command, the question, and the answer.

Command:	Move your pallet further right.
Question:	What do the words 'your pallet' refer to?
Answer:	Your pallet refers to the pallet you are currently carrying.
Command:	Move closer to it.
Question:	What does the word 'it' refer to?
Answer:	It refers to the empty truck trailer.
Command:	Take the pallet and place it on the one to the left.
Question:	What do the words 'the one' refer to?
Answer:	The one refers to the empty trailer.
Command:	Place the pallet just to the right of the other pallet.
Question:	What do the words 'the pallet' refer to?
Answer:	The wooden crate that the merchandise sits on top of.

Figure 4. : Sample commands, questions, and answers from the corpus.

Resolving linguistic coreferences involves identifying linguistic constituents that refer to the same entity in the external world. For example, in the command, "Pick up the tire pallet and put it on the truck," the noun phrases "the tire pallet" and "it" refer to the same physical object in the external world, or we can also say they *corefer*. Coreference resolution is a well-studied problem in computational linguistics (Jurafsky & Martin, 2008). Although there are several existing software packages to address this problem, most were developed for large corpora of newspaper articles and generalized poorly to language in our corpus. Instead, we created a coreference system that was trained on language from our corpus. Following typical approaches to coreference resolution (Stoyanov et al., 2010), our system consists of a classifier to predict coreference between all pairs of noun phrases in the language combined with a clustering algorithm that enforces transitivity and finds antecedents for all pronouns. For the pair-wise classifier we used a log-linear model that uses bag-of-words features. The model was trained using an annotated corpus of positive and negative pairs of coreferences. We set the classification threshold of the model to 0.5 so that it chooses the result with the most probability mass. Once coreferring variables have been identified, a merging algorithm creates a single unified grounding graph. The coreference resolution algorithm identifies pairs of variables γ in the grounding graph that corefer, and the merging algorithm combines all pairs of coreferring variables. Figure 2 shows a merged graph created from a command, a question, and an answer.

The coreference algorithm is used to merge the information from the open-ended answer to a reset or "What do you mean by 'X'?" question, since answers to both types of questions introduce new noun phrases that must be understood. However, in the case of yes-or-no questions the system has already identified the word or words about which to ask, and the answer provides no new language to be merged so language-based coreference is not needed.

For yes-or-no questions, we incorporate a special factor with local probability of 0 or 1. This

probability can be written as $p(\phi_i|\lambda_i, \gamma_{i_1}, \gamma_{i_2})$, where γ_{i_1} is the grounding variable in the graph corresponding to the original command about which the question was asked, and γ_{i_2} is a grounding variable with value fixed to the grounding g about which the yes-or-no question was asked. In the event of a "yes" answer, this factor's probability is 1 if $\gamma_{i_1} = \gamma_{i_2}$ and 0 otherwise, while in the event of a "no" answer, the factor's probability is 1 if $\gamma_{i_1} \neq \gamma_{i_2}$ and 0 otherwise.

4. Results

We used two datasets to evaluate the system. To focus on commands where questions will have a large impact, we used a corpus of 21 manually created commands given to a simulated robotic forklift (the AMBIGUOUS corpus). These commands were designed to be ambiguous in order to provide an opportunity for clarifying questions and answers. In addition, we used a second larger dataset of natural language commands (the FULL corpus), generated by annotators on Amazon Mechanical Turk and described more fully by Tellex, Kollar, Dickerson, Walter, Banerjee, A., Teller, S., & Roy (2011). This dataset consists of commands given in more complex environments and is much more challenging. To collect commands, we asked annotators to watch a video of the robot carrying out an action. The annotators were then asked to write a natural language command they would give to an expert human forklift operator to ask for performance of the action in the video. We assessed the end-to-end performance of the question-asking framework toward increasing the number of correctly grounded concrete noun phrases and correctly generated robot actions. We used ground-truth parses in all of our experiments.

4.1 Asking Questions

First, we assessed the performance of the system at using answers to questions to disambiguate ambiguous phrases in each corpus. To make this assessment, we needed a corpus of questions and answers for each of the three types of questions. For yes-or-no questions, the system simulated correct answers by testing if the grounding referred to in the yes-or-no question matched the annotated grounding. We then collected answers to reset and targeted questions on Amazon Mechanical Turk (AMT).

During AMT data collection, multiple user-generated commands were collected for each video of the robot performing a given action. These additional commands were used as the answers to reset questions, since they represented the same action described in different words. For targeted questions, we generated a question for each concrete noun phrase[4] in the corpus, and then collected answers to those questions from Mechanical Turk. For example, for a command like, "Take the pallet and place it on the trailer to the left," the question-generation algorithm could ask about "the pallet," "it," or "the trailer to the left." By asking for answers out of all concrete noun phrases in the dataset in advance, we can compare different question selection strategies offline without collecting new data. To collect an answer to a targeted question, we showed annotators a natural language command directing the robot to perform an action in the environment such as, "Pick up the pallet," paired with a question such as, "What do you mean by 'the pallet'?" Then annotators saw a video of the simulated robot performing the action sequence, such as picking up a specific tire pallet in the environment. We instructed them to provide an answer to the question in their own words, assuming that what they saw happening in the video represented the intended meaning of the command. We collected two answers from different annotators for each question. Example commands, questions, and answers from the corpus appear in Table 4.

To measure the performance of the system, we report (a) the fraction of correctly grounded concrete noun phrases in the original command and (b) the fraction of commands for which inference

[4] A concrete noun phrase is one which refers to a specific single object in the external world. "The skid of tires" is a concrete noun phrase, while "your far left-hand side" is not.

generated an action sequence matching that from the video (Tables 1 and 2). A noun phrase such as "the skid of tires" is correct if the inference maps it to the same tire pallet that the human user referenced. It is incorrect if the inference maps it to some other object, such as a trailer. Correctness of an action depends on manipulating (picking up, putting down, and moving) the same objects through the same general points in space.

We evaluate our system in several different conditions, using both automatic coreference resolution and oracle coreference resolution for targeted and reset questions. As baselines, we present the performance using only information from the commands without asking any questions, as well as performance when asking a question about each concrete noun phrase. This baseline is equivalent to the system used by Tellex, Kollar, Dickerson, Walter, Banerjee, A., Teller, S., & Roy (2011). The baseline results in Tables 1 and 2 show that the system realized a large improvement in performance when using information from commands, questions, and answers as compared to information from the commands alone.

Our overall accuracy in understanding commands is low compared to previous approaches to following commands (Tellex, 2010; Matuszek, Fox, & Koscher, 2010; MacMahon, Stankiewicz, & Kuipers, 2006). However these previous approaches were evaluated in the domain of natural language route directions. The state space for a movement task is smaller than movement and manipulation, where the robot can move not only itself but also other objects in the environment, which explains the lower performance.

As a control, we also present a random metric for question selection. In the case of a targeted question, this consisted of choosing a concrete object grounding variable at random about which to ask. For a yes-or-no question, we generated a list of all possible pairings of variables and groundings and selected pairs at random from that list. We report the mean and 95% confidence interval of object correctness for 10 runs of the random metric in each case, except targeted questions on the FULL corpus for which only five runs were performed for each case. Due to the time expense of evaluating command correctness, which must be manually annotated, for the random metric we report it for only one randomly-sampled run.

4.2 Yes-or-No Questions

When asking yes-or-no questions, the system showed a marked improvement in the accuracy of both concrete noun objects and robot actions. These improvements held across both corpora, the AMBIGUOUS commands corpus in Table 1 and the FULL natural-language corpus shown in Table 2.

4.2.1 Object Correctness Results from the AMBIGUOUS corpus in Table 1 show that Metric 2 (Entropy) and Metric 3 (Event Entropy) performed as well as or better than random selection at improving the number of correctly grounded objects. This difference is much more clear in the FULL corpus of natural-language commands as shown in Table 2, where the more complex environment allowed many more possible questions and made correct question selection more difficult. In this larger corpus, Metric 2 outperformed all other metrics at correctly binding object grounding variables, resulting in an overall improvement from 55% of objects correctly identified in the command only, to 77% after two yes-or-no questions, and 82% after three questions. In contrast, randomly selecting questions resulted in only 65% of objects correctly identified even after three questions were asked and answered. These results can also be seen in Figure 5a.

4.2.2 Event Correctness Metric 3 (Event Entropy) was designed specifically to minimize the uncertainty of the robot's action sequence, and our results demonstrate that it achieved that goal. Metric 3 proved effective at improving event correctness in the smaller, AMBIGUOUS corpus (Table 1), outperforming all other metrics for two and three questions and trailing only Metric 2 with one yes-or-no question. Similar to the results reported in Section 4.2.1, the results from

the FULL corpus in Table 2 show a clearer distinction between the metrics, as demonstrated in Figure 5b. In the FULL corpus, using Metric 3 to select yes-or-no questions resulted in more correctly generated robot actions than any other metric for one, two, and three questions. This performance is particularly striking because Metric 3 resulted in slightly fewer correctly grounded objects than Metric 2 (Entropy), but still managed to outperform Metric 2 in event correctness on the FULL corpus. Since Metric 3 focuses its questions on objects that are most critical to generating the correct action rather than greedily choosing the most uncertain object, it is able to most effectively improve the accuracy of the robot's actions.

For example in one command, the robot was presented with four pallets and told, "Drive to the leftmost pallet of tires and pick it up off the ground." Using Metric 2 (Entropy), the robot determined that the most uncertain concrete noun phrase was "the ground" and asked if those words referred to one of the pallets in its environment. The answer, "No," resulted in an incorrect binding for "the ground" being avoided but did not allow the robot to generate the correct action, as was observed when the robot still traveled to the wrong pallet. Using Metric 3 (Event Entropy), by contrast, the robot asked if the word "it" referred to a particular pallet. The robot received another "No" answer, which allowed it to choose the correct pallet to pick up from the remaining pallets.

4.3 Targeted Questions

Targeted questions showed little effect on the accuracy of objects or events on the larger corpus of 81 commands. Our results in Table 2 show no consistent improvement in performance from Metrics 1 or 2 or the random metric. Oracle coreference merging within each command alone did result in a slight increase of object accuracy over automatic merging or no merging, from 55% to 57%, but the performance after question-asking was still much lower than we observed with yes-or-no questions. In order to determine whether this failure was the result of the performance of the model, or whether the commands and environments used simply did not present opportunities for productive targeted questions, we repeated the evaluation on the deliberately ambiguous corpus.

This ambiguous corpus contains shorter commands which provide less initial information, but it is also set in simpler environments, resulting in a similar level of command-only accuracy. However, the simple commands provide much better opportunities for the robot to gain new and useful information from open-ended responses. The results of this evaluation can be seen in Table 1, which shows dramatic accuracy improvements from one, two, and three targeted questions, as well as substantial performance differences between the three question selection metrics. When asking one targeted question and using automatic coreference, Metric 2 (Entropy) slightly outperformed random question selection but did no better than Metric 1 (Confidence). When asking a second question, Metric 2 significantly outperformed random selection and Metric 1. However, asking three questions with automatic coreference resulted in slightly worse performance of the system since opportunities for errors in coreference resolution rose as more questions and answers added additional nouns phrases to be merged.

To demonstrate the results of the question-selection process independent of a particular coreference algorithm, we also present targeted question results from Metric 1 and Metric 2 using oracle coreference, in which the ground-truth information about the mapping between the linguistic constituents and groundings in the environment is used to identify the correct variables to merge. We show a significant improvement in object accuracy using oracle coreference, as it eliminated errors caused by the automatic resolver (1) merging variables that did not refer to the same object or (2) failing to merge those that did. Using oracle coreference in the AMBIGUOUS corpus, one open-ended answer from a human user was sufficient to achieve 79% accuracy in binding object variables using Metric 1 and 82% accuracy using Metric 2. With just two questions asked, both Metric 2 and the random question selection achieved an object accuracy of 92%. With three targeted questions, the

results from all three metrics converged. This result is not surprising–three questions per command represents approximately three-fourths of all available questions, so all three metrics resulted in the many of the same questions being asked.

4.4 Reset Questions

Reset questions, in which the robot asked the user to rephrase the entire command, generated some improvement in accuracy with one answer but saw no further improvement after two or three answers. In the AMBIGUOUS corpus, using oracle coreference, one reset question raised object accuracy from 57% to 64% but had no effect on event accuracy. Additional reset questions had no effect on accuracy. Using automatic coreference, we observed no improvement in event or object accuracy.

The limited success of reset questions on the AMBIGUOUS corpus is not surprising, given the way that corpus was constructed and the way question-asking was implemented. As explained in Section 4.1, we used additional commands from the same robot action video as reset question answers. In the AMBIGUOUS corpus, these additional commands were, by construction, intentionally ambiguous and thus offered little additional information. For example, for the original command, "Back up and head over to it," the robot received a reset answer of, "Move closer to it," which provided no additional information about what "it" was. By contrast, when the robot asked a targeted question, "What does the word 'it' refer to?" the answer received from a human user was, "It refers to the empty trailer to your far left-hand side," which was sufficiently clear to allow the robot to correctly identify the object in question.

On the full corpus, with commands not designed to be ambiguous, reset performance with one question was somewhat better. Using oracle coreference, one reset question was sufficient to raise object accuracy from 57% to 64% and event accuracy from 31% to 34%. However, additional reset questions reduced accuracy, even below the levels seen from just the command alone. Part of the reason for this can again be seen from the particular answers which were received. For example, the command, "Lift the box pallet on the right to the truck to the left truck," [sic] received the following three answers when three reset questions were asked:

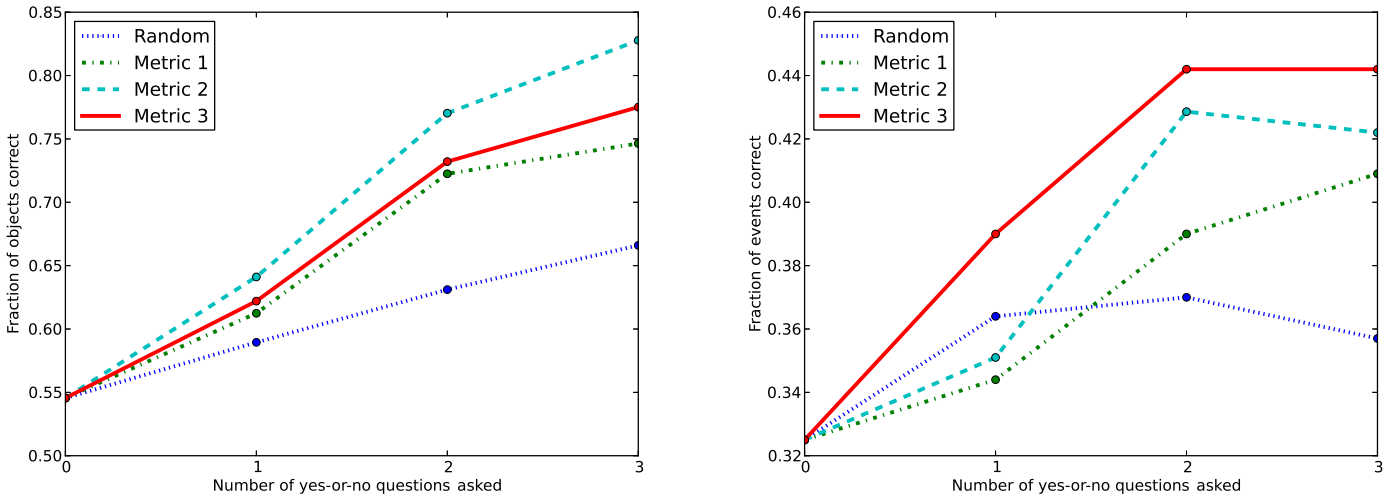
- "Take the pallet of boxes in the middle and place them on the trailer on the left."
- "Pick up the pallet of boxes directly in front of you and drive left to the platform, then set the pallet on top."
- "Pick up the middle skid of boxes and load it onto the trailer."

Between the answers and the command, the box pallet's location is identified as being "on the right" twice, "in the middle" twice, and "directly in front of you" once, even though all four commands were generated by users watching the same video. Unsurprisingly, that box pallet was not correctly identified after three reset questions.

4.5 Challenges

Our framework provides the first steps toward an information-theoretic approach for enabling the robot to ask questions about objects in the environment. Failures occurred for a number of reasons. Annotators providing answers generally did so in good faith, but sometimes those answers were not useful to the robot. For example, one user answered the question, "What do the words 'the pallet' refer to?" with a definition of the pallet ("The wooden crate that the merchandise sits on top of") rather than specifying which pallet was being referenced (e.g., something like, "the pallet with tires.") Other failures occurred in more complex environments because the robot failed to understand the disambiguating answer, as in "the object on the far left," when the system did not have a good model of left versus right. Strategies for improving the model include adding more features, as well as collecting larger datasets for training. For example, the problem of left versus

right involves introducing features that capture the frame of reference being used by the speaker. A second problem is the space of possible actions that the robot can take may not contain any correct action. Increasing the size of the action space could lead to improvement, but may also cause inference to take a long time. We are actively pursing new learning algorithms for learning model parameters using less supervision so that larger datasets may be used without requiring annotation. Yes-or-no questions alleviated some of these problems by ensuring that the answer would be correct and easily understood by the system. The yes-or-no questions also proved to be the most effective at improving object and event accuracy on the FULL corpus, for which the targeted questions were not helpful.



(a) (b)

Figure 5. : The results of using each metric for selecting yes-or-no questions for (a) object accuracy and (b) event accuracy. Metric 2 (Entropy) performs best at object correctness, but Metric 3 (Event Entropy) consistently results in the highest fraction of correct robot actions. These data are taken from the FULL corpus of 81 natural-language commands.

5. Related Work

Many have created systems that exploit the compositional structure of language in order to follow natural language commands (Dzifcak, Scheutz, Baral, & Schermerhorn, 2009; MacMahon, Stankiewicz, & Kuipers, 2006; Winograd, 1971). Previous probabilistic approaches have used generative and discriminative models for understanding route instructions but did not make interactive systems that can use dialog to resolve ambiguity (Kollar, Tellex, Roy, & Roy, 2010; Matuszek, Fox, & Koscher, 2010; Matuszek, Herbst, Zettlemoyer, & Fox, 2012; Shimizu & Haas, 2009; Tellex, Kollar, Dickerson, Walter, Banerjee, A., Teller, S., & Roy, 2011; Vogel & Jurafsky, 2010). Other work in semantic parsing uses supervised data consisting of sentences paired with logical form to learn word meanings (Kwiatkowski, Zettlemoyer, Goldwater, & Steedman, 2010; Piantadosi, Goodman, Ellis, & Tenenbaum, 2008; Wolfie, 2003; Wong & Mooney,

		Command Only		One Question		Two Questions		Three Questions	
		Objects	Actions	Objects	Actions	Objects	Actions	Objects	Actions
Yes-or-No	Random	55%	31%	66 ± 2.9%	34%	73 ± 3.1%	43%	79 ± 2.6%	37%
	Metric 1 (Confidence)	55%	31%	63%	40%	76%	40%	82%	40%
	Metric 2 (Entropy)	55%	31%	**68**%	**46**%	74%	43%	89%	43%
	Metric 3 (Event Entropy)	55%	31%	66%	40%	**84**%	**46**%	**92**%	**46**%
Targeted, Oracle Coreference	Random	55%	31%	77 ± 2.4%	43%	**92 ± 2.9**%	46%	94 ± 2.1%	46%
	Metric 1 (Confidence)	55%	31%	79%	43%	87%	46%	**95**%	**49**%
	Metric 2 (Entropy)	55%	31%	**82**%	**46**%	**92**%	**49**%	**95**%	**49**%
Targeted, Automatic Coreference	Random	55%	31%	57 ± 0.8%	34%	58 ± 2.4%	**37**%	61 ± 2.0%	**37**%
	Metric 1 (Confidence)	55%	31%	**58**%	34%	61%	34%	58%	34%
	Metric 2 (Entropy)	55%	31%	**58**%	34%	**66**%	**37**%	**63**%	**37**%
Reset, Oracle Coreference		55%	31%	**61**%	31%	**61**%	31%	**61**%	31%
Reset, Automatic Coreference		55%	31%	55%	31%	55%	31%	55%	31%

Table 1: Object and action accuracy of the robot's inference after asking the chosen number and type of questions. These results were taken from the AMBIGUOUS corpus of 21 commands, deliberately designed to be ambiguous. Results from the metric with the highest accuracy in each category are shown in bold.

		Command Only		One Question		Two Questions		Three Questions	
		Objects	Actions	Objects	Actions	Objects	Actions	Objects	Actions
Yes-or-No	Random	55%	33%	59 ± 0.6%	36%	63 ± 1.2%	37%	67 ± 0.8%	36%
	Metric 1 (Confidence)	55%	33%	61%	34%	72%	39%	75%	41%
	Metric 2 (Entropy)	55%	33%	**64**%	35%	**77**%	43%	**83**%	42%
	Metric 3 (Event Entropy)	55%	33%	62%	**39**%	73%	**44**%	77%	**44**%
Targeted, Oracle Coreference	Random	57%	31%	**58 ± 2.8**%	30%	**59 ± 1.6**%	**31**%	57 ± 1.6%	**30**%
	Metric 1 (Confidence)	57%	31%	56%	**31**%	56%	29%	56%	29%
	Metric 2 (Entropy)	57%	31%	**58**%	**31**%	58%	27%	**58**%	29%
Targeted, Automatic Coreference	Random	55%	31%	55 ± 1.2%	29%	**55 ± 1.6**%	29%	53 ± 2.3%	27%
	Metric 1 (Confidence)	55%	31%	53%	31%	50%	29%	50%	29%
	Metric 2 (Entropy)	55%	31%	**57**%	**32**%	53%	29%	**54**%	**30**%
Reset, Oracle Coreference		**57**%	31%	**64**%	**34**%	**54**%	29%	**52**%	27%
Reset, Automatic Coreference		55%	31%	54%	30%	49%	29%	48%	27%

Table 2: Object and action accuracy of the robot's inference after asking the chosen number and type of questions. These results were taken from the FULL corpus of 81 natural language commands. Results from the metric with the highest accuracy in each category are shown in bold.

2007; Zettlemoyer & Collins, 2005); some of these approaches have been applied to robotics (Chen & Mooney, 2011; Matuszek, Herbst, Zettlemoyer, & Fox, 2012). However, because word meanings are not physically grounded, our entropy-based approach can not be leveraged to generate questions. Our work instead focuses on using an induced probabilistic model over natural language commands and groundings in order to engage in dialog with the human user, asking targeted questions and incorporating information from the answer.

Others have created robotic systems that interact using dialog (Cantrell et al., 2012; Dzifcak, Scheutz, Baral, & Schermerhorn, 2009; Hsiao et al., 2008; Skubic et al., 2004). Bauer et al. (2009) built a robot that can find its way through an urban environment by interacting with pedestrians using a touch screen and gesture recognition system. Fong, Nourbakhsh, & Dautenhahn (2003) developed a collaborative control system in which robots asked human partners questions via a PDA interface. Our approach differs in that it focuses on simple, structured dialogs but is able to understand language from untrained users rather than a predefined, fixed vocabulary of answers. Furthermore, it chooses targeted questions using an information-theoretic metric based on the robot's model of the external world.

Existing work in dialog systems (Doshi & Roy, 2008; Roy, Pineau, & Thrun, 2000; Williams & Young, 2007b,a; Young, 2006) use MDP and POMDP models with a fixed, predefined state space to represent the user's intentions. These frameworks are able to integrate parallel dialog state hypotheses, confidence scores, and automated planning into a single decision-theoretic framework. However, existing approaches use a predefined state-action space for the dialog system and user goals. Our approach, in contrast, dynamically defines dialog actions for the system (e.g., questions it could ask) from the words in the user's command and contextual information the surrounding environment. Ultimately, we envision integrating our approach into a POMDP planning framework with a dynamically defined state space, capturing the best of both approaches.

Researchers in multimodal interaction have studied voice interfaces compared to other interaction modalities for spatial tasks. Cohen & Oviatt (1995) report that speech interfaces are useful when the hands and eyes are otherwise engaged and when there is limited availability of keyboards or screens. Robots, which operate in unstructured, real-world environments fit these scenarios perfectly. However, there is not consensus that human-robot interfaces should be built around natural language due to the challenges in building dialog interfaces in changing social contexts (Fong, Nourbakhsh, & Dautenhahn, 2003; Severinson-Eklundh, Green, & Hüttenrauch, 2003). The aim of our work is to develop robust natural language dialog systems so that robots can interact with people flexibly using language.

6. Conclusion

In this paper, we presented results for a robot dialog understanding system based on a probabilistic graphical model that factors according to the structure of language. The robot is able to ask the human user questions about parts of a command that it had failed to understand and incorporate information from an open-ended space of answers into its model, iteratively improving its confidence and accuracy.

Our next steps are to scale the algorithm to more complex dialogs, by allowing the robot to ask more flexible questions and combine different types of questions for a single command. Developing a model to predict the effect of open-ended answers to questions will allow event entropy to be used for choosing targeted questions in addition to yes-or-no questions. Integrating nonverbal backchannels and gestures into the framework as new types of factors in the grounding graph remains an open problem. We aim to extend the framework to support active learning, enabling the robot to learn new word meanings based on answers it has received to questions and to extend the dialog system beyond noun-phrase and reset questions. The entropy metric for noun phrases

could be extended to directly measure the entropy of the distribution of possible robot actions, and a simulation of the most probable action could be presented to the user for yes-or-no confirmation. This would likely be particularly useful in the case of a command which consists of multiple events with varying certainty, such as traveling to a location and then picking up an object. In addition, verb learning could be further improved by dialog through requests for demonstration of an action. If the robot had low certainty about its action but high certainty about all of the objects in the command, it could request demonstration of the commanded action, which would provide additional training data for grounding verbs to actions.

7. Acknowledgments

This work was sponsored by Dr. Jon Borenstein and the U.S Army Research Laboratory under the Robotics Collaborative Technology Alliance, by Dr. Tom McKenna and the Office of Naval Research under MURI N00014-07-1-0749, by DARPA via ONR contract #HR0011-11-2-0008, by Battelle under Dr. Robert Carnes and Scott Sheaf, and by the Fannie and John Hertz Foundation. Their support is gratefully acknowledged. We would like to thank Javier Velez, Ashis Banerjee and Joshua Joseph for helpful discussions about this work. We also thank Seth Teller and the Agile Robotics for Logistics team for use of the robotic forklift.

References

Bauer, A., Klasing, K., Lidoris, G., Mühlbauer, Q., Rohrmüller, F., Sosnowski, S., et al. (2009, April). The Autonomous City Explorer: Towards natural human-robot interaction in urban environments. *International Journal of Social Robotics*, *1*(2), 127–140.

Cantrell, R., Talamadupula, K., Schermerhorn, P., Benton, J., Kambhampati, S., & Scheutz, M. (2012). Tell me when and why to do it!: Run-time planner model updates via natural language instruction. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (pp. 471–478). New York, NY, USA: ACM.

Chen, D. L., & Mooney, R. J. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (pp. 859–865).

Cohen, P., & Oviatt, S. (1995). The role of voice input for human-machine communication. In *Proceedings of the National Academy of Sciences* (Vol. 92, pp. 9921–9927). National Academy Sciences.

Doshi, F., & Roy, N. (2008). Spoken language interaction with model uncertainty: An adaptive human-robot interaction system. *Connection Science*, *20*(4), 299–319.

Dzifcak, J., Scheutz, M., Baral, C., & Schermerhorn, P. (2009). What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *IEEE International Conference on Robotics and Automation* (pp. 4163–4168).

Fong, T., Nourbakhsh, I., & Dautenhahn, K. (2003). A survey of socially interactive robots. *Robotics and autonomous systems*, *42*(3), 143–166.

Hsiao, K., Tellex, S., Vosoughi, S., Kubat, R., & Roy, D. (2008). Object schemas for grounding language in a responsive robot. *Connection Science*, *20*(4), 253–276.

Jackendoff, R. S. (1985). *Semantics and cognition* (Vol. 8). MIT Press.

Jurafsky, D., & Martin, J. H. (2008). *Speech and language processing* (2nd ed.). Englewood Cliffs, New Jersey: Pearson Prentice Hall.

Kollar, T., Tellex, S., Roy, D., & Roy, N. (2010). Toward understanding natural language directions. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (pp. 259–266).

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., & Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 1223–1233).

MacMahon, M., Stankiewicz, B., & Kuipers, B. (2006). Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (pp. 1475–1482).

Marneffe, M. de, MacCartney, B., & Manning, C. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)* (pp. 449–454). Genoa, Italy.

Matuszek, C., Fox, D., & Koscher, K. (2010). Following directions using statistical machine translation. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (pp. 251–258).

Matuszek, C., Herbst, E., Zettlemoyer, L., & Fox, D. (2012). Learning to parse natural language commands to a robot control system. In *Proceedings of the International Symposium on Experimental Robotics (ISER).* Quebec City, Canada.

McCallum, A. K. (2002). *MALLET: A machine learning for language toolkit.* http://mallet.cs.umass.edu.

Piantadosi, S., Goodman, N., Ellis, B., & Tenenbaum, J. (2008). A Bayesian model of the acquisition of compositional semantics. In *Proceedings of the Thirtieth Annual Conference of the Cognitive Science Society* (pp. 1620–1625).

Rosenthal, S., Veloso, M., & Dey, A. K. (2011). Learning accuracy and availability of humans who help mobile robots. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (pp. 1501–1506).

Roy, N., Pineau, J., & Thrun, S. (2000). Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th annual meeting of the association for computational linguistics (ACL-2000).*

Severinson-Eklundh, K., Green, A., & Hüttenrauch, H. (2003). Social and collaborative aspects of interaction with a service robot. *Robotics and Autonomous Systems*, *42*(3), 223–234.

Shimizu, N., & Haas, A. (2009). Learning to follow navigational route instructions. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence* (pp. 1488–1493).

Simeonov, D., Tellex, S., Kollar, T., & Roy, N. (2011). Toward interpreting spatial language discourse with grounding graphs. In *RSS Workshop on Grounding Human-Robot Dialog for Spatial Tasks.* Los Angeles, CA.

Skubic, M., Perzanowski, D., Blisard, S., Schultz, A., Adams, W., Bugajska, M., & Brock, D. (2004). Spatial language for human-robot dialogs. *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *34*(2), 154–167.

Stoyanov, V., Cardie, C., Gilbert, N., Riloff, E., Buttler, D., & Hysom, D. (2010, April). *Reconcile: A coreference resolution research platform* (Tech. Rep.). Cornell University.

Tellex, S. (2010). *Natural Language and Spatial Reasoning*. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA.

Tellex, S., Kollar, T., Dickerson, S., Walter, M., Banerjee, A., Teller, S., & Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI).* San Francisco, CA.

Tellex, S., Kollar, T., Dickerson, S., Walter, M. R., Banerjee, A. G., Teller, S., & Roy, N. (2011). Approaching the symbol grounding problem with probabilistic graphical models. *AI Magazine*, *32*(4), 64–76.

Tellex, S., Thaker, P., Deits, R., Kollar, T., & Roy, N. (2012, July). Toward information theoretic human-robot dialog. In *Proceedings of Robotics: Science and Systems.* Sydney, Australia.

Thompson, C. A., & Mooney, R. J. (2003). Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research*, *18*, 1–44.

Vogel, A., & Jurafsky, D. (2010). Learning to follow navigational directions. In *Proceedings of the Association for Computational Linguistics* (pp. 806–814).

Williams, J. D., & Young, S. (2007a, April). Partially observable Markov decision processes for spoken dialog systems. *Computer Speech & Language*, *21*(2), 393–422.

Williams, J. D., & Young, S. (2007b, September). Scaling POMDPs for spoken dialog management. *IEEE Transactions on Audio, Speech, and Language Processing*, *15*(7), 2116–2129.

Winograd, T. (1971). *Procedures as a Representation for Data in a Computer Program for Understanding Natural Language*. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA.

Wong, Y., & Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for computational linguistics* (Vol. 45, p. 960).

Young, S. (2006). Using POMDPs for dialog management. In *IEEE Spoken Language Technology Workshop* (pp. 8–13).

Zettlemoyer, L. S., & Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence)* (pp. 658–666).

Authors' names and contact information: Stefanie Tellex, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA. Email: stefie10@csail.mit.edu. Pratiksha Thaker, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA. Email: prthaker@csail.mit.edu. Robin Deits, Battelle Memorial Institute, Columbus, Ohio, USA. Email: robin.deits@gmail.com Dimitar Simeonov, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA. Email: mitko@csail.mit.edu. Thomas Kollar: Carnegie Mellon University, Pittsburgh, PA, USA. Email: tkollar@cmu.edu. Nicholas Roy, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA. Email: nickroy@csail.mit.edu.