Toward Information Theoretic Human-Robot Dialog

Stefanie Tellex[1†], Pratiksha Thaker[1†], Robin Deits[‡], Dimitar Simeonov[†], Thomas Kollar[§], Nicholas Roy[†]
[†]MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA
Email: {stefie10, prthaker, mitko, nickroy}@csail.mit.edu
[‡]Battelle Memorial Institute, Columbus, Ohio
Email: robin.deits@gmail.com
[§]Carnegie Mellon University, Pittsburgh, PA
Email: tkollar@cmu.edu

***Abstract*—Our goal is to build robots that can robustly interact with humans using natural language. This problem is challenging because human language is filled with ambiguity, and furthermore, due to limitations in sensing, the robot's perception of its environment might be much more limited than that of its human partner. To enable a robot to recover from a failure to understand a natural language utterance, this paper describes an information-theoretic strategy for asking targeted clarifying questions and using information from the answer to disambiguate the language. To identify good questions, we derive an estimate of the robot's uncertainty about the mapping between specific phrases in the language and aspects of the external world. This metric enables the robot to ask a targeted question about the parts of the language for which it is most uncertain. After receiving an answer, the robot fuses information from the command, the question, and the answer in a joint probabilistic graphical model in the G^3 framework. When using answers to questions, we show the robot is able to infer mappings between parts of the language and concrete object groundings in the external world with higher accuracy than by using information from the command alone. Furthermore, we demonstrate that by effectively selecting which questions to ask, the robot is able to achieve significant performance gains while asking many fewer questions than baseline metrics.**

I. 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 referring expressions that do not map to a unique object in the external world. For 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 person 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 recover from 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 parts of a command, 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 "Pick up the pallet on the truck" in the situation shown in Figure 1, the robot can infer that because there is only one truck in the scene, but two pallets, the phrase "the pallets" is the most ambiguous and ask a question like, "What do you mean by 'the pallet'?" Then it 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.

(a)

```
Move the pallet from the truck.

Remove the pallet from the back of the truck.

Offload the metal crate from the truck.
```

(b)

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

Previous approaches to robotic question-asking do not directly map between natural language and perceptually-grounded aspects of the external world or incorporate additional information from free-form natural language answers in order to disambiguate the command. [3, 11, 1]. As a result, the robot cannot take advantage of its model of the environment to determine the most ambiguous parts of an arbitrary natural language command and identify a question to ask.

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

In order to derive an expression for the robot's uncertainty about groundings in the external world, our approach builds on the Generalized Grounding Graph (G^3) framework [18, 17]. 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. However, this factorization requires introducing a new *correspondence* variable which leads to difficulties when estimating the entropy over grounding variables. In this paper we derive a metric based on entropy using the G^3 framework. The robot uses this metric to identify the most uncertain random variables in the model 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 model, the robot infers the best set of groundings corresponding to the command, the question, and the answer.

We evaluate the system by collecting answers to questions created by the robot using crowdsourcing. We demonstrate that the system 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 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 [14].

II. Background

We briefly review grounding graphs, which were introduced by [18], giving special attention to the motivation for the correspondence variable, Φ. The correspondence variable makes it possible to efficiently train the model using local normalization at each factor but complicates the calculation of entropy described in Section III-A.

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. The aim is to find the most probable groundings $\gamma_1 \dots \gamma_N$ given the language Λ and the robot's model of the environment m:

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

For brevity, we omit m from future equations. Groundings are the specific physical concept that is meant by the language and can be objects (e.g., a truck or a door), places (e.g., a particular location in the world), paths (e.g., a trajectory through the environment), or events (e.g., a sequence of robot actions).

To learn this distribution, one standard approach is to factor it based on certain independence assumptions, then train models for each factor. Natural language has a well-known compositional, hierarchical argument structure [6], and a promising approach is to exploit this structure in order to factor the model. However, if we define a directed model over these variables, we must assume a possibly arbitrary order to the conditional γ_i factors. For example, for a phrase such as "the tire pallet near the other skid," we could factorize 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.

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 grounding γ_i. 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.

1) Inference: In order to use the G^3 framework for inference, we want to infer the groundings $\gamma_1 \dots \gamma_N$ that maximize the distribution

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

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

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

We assume that Λ and $\gamma_1 \dots \gamma_N$ are independent when Φ is not known, yielding:

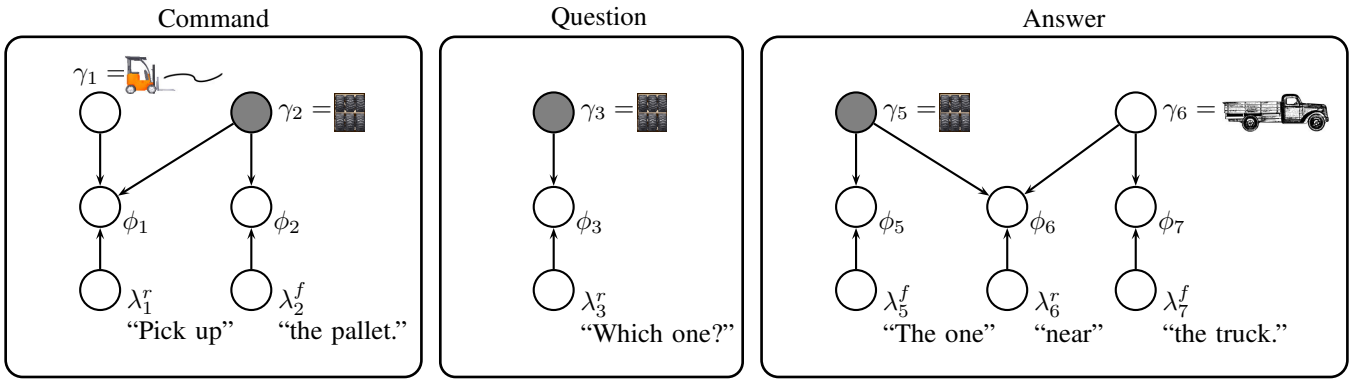
$$\underset{\gamma_1 \dots \gamma_N}{\operatorname{argmax}} \, p(\Phi | \Lambda, \gamma_1 \dots \gamma_N) p(\Lambda) p(\gamma_1 \dots \gamma_N) \tag{6}$$

This independence assumption may seem unintuitive, but it is justified because the correspondence variable Φ breaks the dependency between Λ and $\gamma_1 \dots \gamma_N$. If we do not know whether $\gamma_1 \dots \gamma_N$ correspond to Λ, we assume that the language does not tell us anything about the groundings.

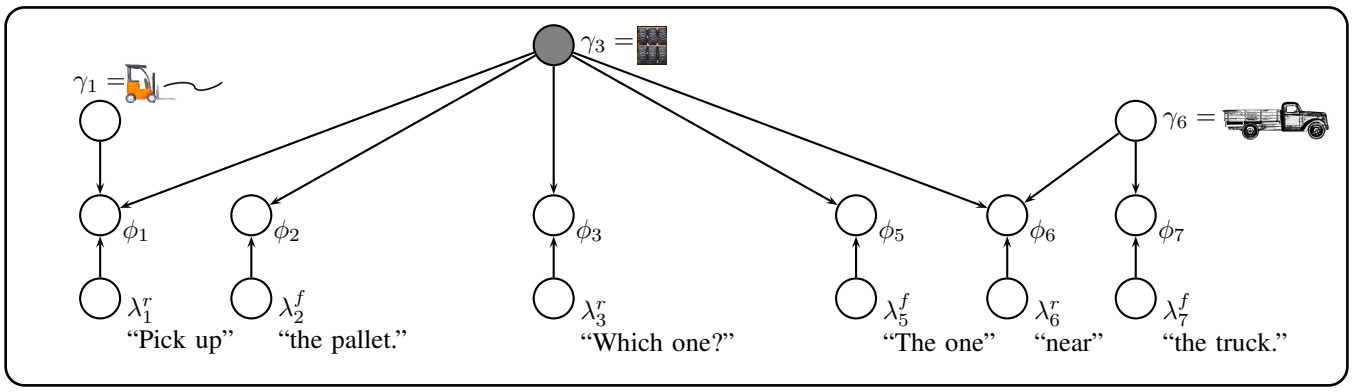
Finally, we assume a constant prior on $\gamma_1 \dots \gamma_N$ and ignore $p(\Lambda)$ since it does not depend on $\gamma_1 \dots \gamma_N$, leading to:

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

The G^3 framework described by [18] trains the model in a discriminative fashion. However, it is not a conventional conditional random field in that the correspondence vector $\Phi = True$ is observed, and the conditioning variables γ_i are hidden, preventing the use of standard inference techniques. To compute the maximum value of the objective in Equation 7, the system performs beam search over $\gamma_1 \dots \gamma_N$, computing



(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.

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

the probability of each assignment from Equation 7 to find the maximum probability assignment. 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. This approach is valid because, given our independence assumptions, $p(\Phi|\Lambda, \gamma_1 \dots \gamma_N)$ corresponds to the joint distribution over all the variables given in Equation 5.

In order to perform beam search, we factor the model according to the hierarchical, compositional linguistic structure of the command:

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

This factorization can be represented graphically; we call such a graphical model the *grounding graph* for a natural language command. We can draw it as a factor graph, but in this paper we represent it as a directed graphical model to emphasize the role of the correspondence variable and independence assumptions in the factorization. The directed model for the command "Pick up the pallet" appears in Figure 2. The λ variables correspond to language; the γ variables correspond to groundings in the external world, and the ϕ variables are $True$ if the groundings correspond to the language, and $False$ otherwise.

2) Training: We learn model parameters from a corpus of labeled examples. The G^3 framework assumes a log-linear parametrization with feature functions f_j and feature weights μ_j:

$$p(\Phi|\Lambda, \gamma_1 \dots \gamma_N) = \prod_i \frac{1}{Z} \exp(\sum_j \mu_j f_j(\phi_i, \lambda_i, \gamma_{i_1} \dots \gamma_{i_k})) \quad (8)$$

Features map between words in the language and corresponding groundings in the external world. For example, features include object class, whether one grounded object is physically supported by another grounded object, or whether the robot is approaching or moving away from a landmark object. The training set consists of an aligned, parallel corpus of language paired with positive and negative examples of groundings in the external world. The alignment annotations consist of a mapping between each natural language constituent and a corresponding grounding in the external world. We use the corpus and alignment annotations described in [18].

III. Technical Approach

When faced with a command, the system extracts grounding graphs from the natural language input and performs inference to find the most likely set of values for the grounding variables

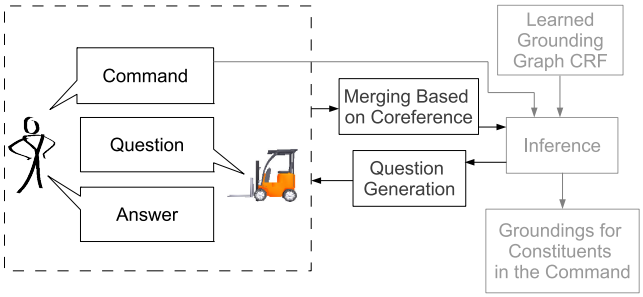


Fig. 3: System diagram. Grayed out blocks show pre-existing components; black parts show the question-asking feedback system new to this paper.

$\gamma_1 \dots \gamma_N$. Next, it identifies the most uncertain grounding variable γ_j and asks a question about that variable, described in Section III-A. After receiving an answer from a human partner, the robot merges grounding graphs from the command, question, and answer into a single graphical model, described in Section III-B. Finally, it 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 dataflow in the system. Grayed-out blocks in the figure show pre-existing components, not novel to this paper. Black blocks show the question-asking feedback system that is the contribution of this work.

A. Generating a Question

Our approach to asking questions is to first identify grounding variables whose values are most uncertain, then generate a question to try to disambiguate the value of that variable. The system iteratively asks questions about the most uncertain grounding variable γ_j until it is sufficiently confident about having inferred the correct groundings.

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:

$$\underset{\gamma_j}{\text{argmin}} \prod_{k \in factors(\gamma_j)} p(\phi_k | \gamma_1 \dots \gamma_N, \Lambda) \tag{9}$$

If the system was unable to find a high-probability grounding for a variable γ_j, then it could ask a question to collect more information. We refer to this approach as **Metric 1**. [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 grounding variables, then ask a question about the variable with maximum entropy. We begin by defining a family of marginal distributions for each grounding variable γ_j conditioned on Φ and Λ:

$$p(\gamma_j | \Phi, \Lambda) \tag{10}$$

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

$$\underset{j}{\text{argmax}}\, H_{p(\gamma_j|\Phi,\Lambda)}(\gamma_j) \tag{11}$$

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 "Which 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 "Which pallet?"

We can expand the entropy function as:

$$H_{p(\gamma_j|\Phi,\Lambda)}(\gamma_j) = -\sum_{\gamma_j} p(\gamma_j | \Phi, \Lambda) \log p(\gamma_j | \Phi, \Lambda) \tag{12}$$

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

$$p(\gamma_j | \Phi, \Lambda) = \sum_{\gamma_1 \dots \gamma_{j-1}, \gamma_{j+1} \dots \gamma_N} p(\gamma_1 \dots \gamma_N | \Phi, \Lambda) \tag{13}$$

We use Bayes' rule to rewrite it:

$$p(\gamma_j | \Phi, \Lambda) = \sum_{\gamma_1 \dots \gamma_{j-1}, \gamma_{j+1} \dots \gamma_N} \frac{p(\Phi | \gamma_1 \dots \gamma_N, \Lambda) p(\gamma_1 \dots \gamma_N | \Lambda)}{p(\Phi | \Lambda)} \tag{14}$$

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:

$$p(\gamma_j | \Phi, \Lambda) = \sum_{\gamma_1 \dots \gamma_{j-1}, \gamma_{j+1} \dots \gamma_N} \frac{p(\Phi | \gamma_1 \dots \gamma_N, \Lambda) p(\gamma_1 \dots \gamma_N)}{p(\Phi | \Lambda)} \tag{15}$$

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

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

We can 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; we call each sample s_m. Each sample consists of bindings for grounding variable γ_j

[1] We use confidence instead of $H_{p(\Phi|\gamma_1 \dots \gamma_N, \Lambda)}(\Phi)$, since entropy is low when $p(\phi_k | \gamma_1 \dots \gamma_N, \Lambda)$ is either very high or very low probability.

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 be large so that the samples are diverse enough to enable accurate estimate of entropy. We approximate 16 with:

$$\hat{p}(\gamma_j = g|\Phi, \Lambda) = \frac{c(\gamma_j = g, \Phi, \Lambda)}{\sum_x c(\gamma_j = x, \Phi, \Lambda)} \quad (17)$$

where c is

$$c(\gamma_j = g|\Phi, \Lambda) = K \sum_{\{\, s_m | s_m[\gamma_j]=g \,\}} p(\Phi|s_m[\gamma_1] \dots s_m[\gamma_N], \Lambda) \times p(s_m[\gamma_1] \dots s_m[\gamma_N]) \quad (18)$$

We can substitute this equation into Equation 12 to obtain an estimate for the entropy. We refer to this approximation as **Metric 2**.

After identifying a variable to ask about, the robot asks a question using a template-based algorithm. It finds text associated with the grounding variable and 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. In general, answers could take many forms. For example, Figure 5 shows commands, questions generated using the template-based algorithm, along with corresponding answers collected from untrained users.

B. Merging Graphs

Once a question has been chosen and an answer obtained, the robot incorporates information from the answer into its inference process. It 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, question, and answer.

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 *corefer*. Coreference resolution is a well-studied problem in computational linguistics [7]. Although there are several existing software packages to address this problem, most are developed for large corpora of newspaper articles and generalize poorly to language in our corpus. Instead, we created a coreference system which is trained on language from our corpus. Following typical approaches to coreference resolution [16], 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 which uses bag-of-words features. The model is 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 choses 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 γ variables in the grounding graph that corefer; the merging algorithm combines all pairs of coreferring variables. Figure 2 shows a merged graph created from a command, a question, and an answer.

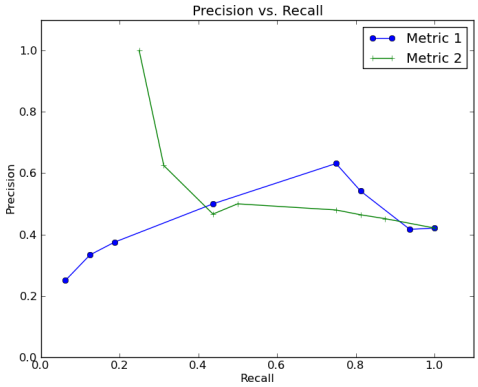


Fig. 4: Precision vs. recall at predicting whether an inferred grounding is incorrect.

C. Deciding When to Ask a Question

Section III-A described how to generate a question in response to a natural language command. However, at a higher level, the robot needs to decide whether to ask a question or take an action. For example, if the robot is very confident about all grounding variables, it would be better to ask no questions at all. If it is uncertain about just one variable, a single question might suffice to disambiguate the command. Or it might ask a question, get an answer that it cannot understand, then choose to ask an additional question about the same grounding variable to receive further clarification. Our approach to this problem is to ask questions until the entropy of the most uncertain variable is below a certain threshold. To avoid going into an infinite loop, we prohibit the robot from asking a question about the same variable more than two times.

IV. Results

To evaluate the system, we use a corpus of 21 manually created commands given to a simulated robotic forklift. The commands were designed to be deliberately ambiguous in order to provide an opportunity for clarifying questions and answers. In Section IV-A, we assess the performance of the Metric 1 (Confidence) and Metric 2 (Entropy) at identifying incorrect grounding variables after inference has been performed for commands in the dataset. Second, we assess the end-to-end performance of the question-asking framework at increasing the number of correctly grounded concrete noun phrases.

A. Predicting Incorrect Examples

To directly assess the performance of the metrics defined in Section III-A, we measure their performance at identifying

grounding variables in the corpus that are incorrect. Figure 4 shows the effect of this process on the commands from the corpus as a precision vs. recall curve. Here, a true positive is an incorrect grounding variable that was predicted to be incorrect; a false positive is a correct grounding variable that was predicted to be incorrect. Metric 1 (Confidence) performs quite poorly, even having positive slope. This is not due to a bug; instead, there is a group of low-confidence grounding variables which are in fact correct. For example, for the command "Lift it from the truck," the factor for "the truck" has high confidence, but "from the truck" has lower confidence. Metric 1 multiplies the two values together, yielding an overall low estimate for confidence. Metric 2, in contrast, gives this variable much lower entropy compared to other commands, since the "from the truck."

B. Asking Questions

Next, we assess the performance of the system at using answers to questions to disambiguate ambiguous phrases in the corpus. To collect a corpus of questions and answers, we first generated questions for each concrete noun phrase in the corpus, then collected answers to those questions using crowdsourcing. 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 answers for 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 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'?" In addition, 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 5.

To measure the performance of the system, we report the fraction of correctly grounded concrete noun phrases in the original command, not including the question and answer. A concrete noun phrase is one which refers to a specific single object in the external world, such as "the skid of tires." An example of a non-concrete noun phrase is "your far left-hand side." A noun phrase such as "the skid of tires" is considered to be correct if the inference maps it to the same tire pallet that the human user referenced. It is considered to be incorrect if the inference maps it to some other object, such as a trailer.

We evaluate our system in several different conditions, using both automatic coreference resolution and oracle coreference resolution. 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. The baseline results in Table I show that the system realizes a large improvement in performance when using information from commands, questions, and answers compared to information from the commands alone.

Next, we assess the performance of the two metrics at selecting questions to ask. We report performance for three conditions: selecting just one question to ask about a command, selecting two questions, and selecting questions until uncertainty is below a specified, hand-tuned threshold. We also compare to selecting a question at random. The system may ask up to two questions about each concrete noun phrase. When asking about a noun phrase for a second time, it generates the same question but receives a different answer. The system could ask a total of 76 possible questions over the 21 commands in the corpus; for each approach we also report the fraction of questions from this space that were actually asked. Table I shows the performance of the system in these conditions.

When asking one question and using automatic coreference resolution, Metric 2 (Entropy) slightly outperforms Metric 1 (Confidence), but does no better than randomly choosing a question to ask. When asking a second question, Metric 2 achieves better accuracy than Metric 1 or random question selection, nearly matching the 71% correct achieved by asking all possible questions. This is despite the fact that it only asks 55% of the available questions. These results show clear improvement from the additional information in the questions and answers, and some advantage from Metric 2 over Metric 1 and random selection of questions. Note when asking a second question, the system may choose to ask again about the previously encountered phrase; it may also choose to ask about another phrase. This mechanism means that if the answer to the first question was ambiguous, the system can recover by asking again to collect more information.

In order to focus on the results of the question-selection process, we repeat the analysis of the two metrics using oracle coreference, which determines linguistic coreference directly from the mapping between constituents in the language and groundings in the real world. This eliminates automatic coreference as a source of error in these results. We see a significant improvement using oracle coreference compared to automatic coreference; this is due to errors in which the automatic resolver merges variables that do not actually refer to the same object.

When using oracle coreference, asking all possible questions shows a dramatic improvement over asking no questions (92% from 58%). With one and two questions being asked, Metric 2 (Entropy) again outperforms Metric 1 (Confidence) and random selection. Notably, question selection with Metric 2 (Entropy) is able to achieve the same 92% accuracy as asking all questions, despite only asking 55% of the questions available to it. Improved coreference resolution could be achieved by training on a larger corpus of examples, as well as adding additional features to the coreference resolver, especially including information from groundings.

Next, we assess the system's performance using the algorithm described in Section III-C in order to decide when to ask a question, in addition to deciding what question to ask. Table II shows the performance of Metric 1 and Metric 2. Metric 2 significantly outperforms Metric 2 while asking only a few more questions. Although it asks less than half of the possible questions, it approaches the performance of the system which asks all possible questions.

As an example of the system's operation, for a command such as "Move the pallet over to it," the entropy according to Metric 2 for the phrase "it" is (1.82) and "the pallet" (1.00). The system identifies "it" as the most uncertain variable, then asks "What does the word 'it' refer to?" The answer was "It refers to the empty trailer to the left of the two pallets." After incorporating the answer into the model and performing inference, the system correctly grounds the phrase "it" to the trailer and computes a new estimate for the entropy. The entropy for "it" has now been reduced, and the robot next asks "What do the words the pallet refer to?" and receives an answer "The pallet is the piece of wood with the orange and grey boxes that is directly in front of the forklift." After this process, the robot has inferred correct values for all grounding variables in the grounding graph for this command.

Finally, we assess the system's performance at producing better action sequences using answers to questions. For each top-level clause in the corpus, we generated a sequence of robot actions and manually assessed whether those actions matched the actions in the original video. When using information from commands only, the system correctly executes 37% of the top level actions. In many cases, for a command such as "pick up the pallet" it does pick up a pallet, but a different one from the original video. In contrast, after incorporating information from the answers to questions, it correctly executes 66% of the commands.

Our framework provides 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. Sometimes, the answer obtained from the human user was not useful. For example, one user answered "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. Our entropy-based approach is limited to types of questions that target specific phrases in the answer. More general algorithms are needed to handle a wider array of dialog strategies, such as asking yes/no' questions, such as "Do you mean this one?" or more open-ended questions such as "Now what?"

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.

Fig. 5: Sample commands, questions, and answers from the corpus.

TABLE I: Performance at Grounding Concrete Noun Phrases

	% Correct, automatic coreference	% Correct, oracle coreference	% Questions Asked
Baselines			
No Questions	47%	58%	0%
All Questions	71%	92%	100%
Asking One Question			
Random	55%	74%	28%
Metric 1 (Confidence)	53%	74%	28%
Metric 2 (Entropy)	55%	79%	28%
Asking Two Questions			
Random	61%	82%	55%
Metric 1 (Confidence)	63%	82%	55%
Metric 2 (Entropy)	68%	92%	55%

V. Related Work

Many have created systems that exploit the compositional structure of language in order to follow natural language commands [21, 9, 4]. 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 [10, 19, 8, 18, 13]. Our work instead focuses on using an induced probabilistic model over natural language commands and groundings in order to incorporate information from questions and answers.

Others have created robotic systems that interact using dialog [5, 15, 2]. Bauer et al. [1] built a robot that can find its way through an urban environment by interacting with pedestrians using a touch screen and gesture recognition system. Our approach differs in that it focuses on simple three-turn dialogs but is able to understand language from untrained users rather than a predefined, fixed vocabulary or fixed types of answers. Furthermore, it chooses targeted questions based on the robot's model of the external world.

Existing work in dialog systems [3, 12, 20, 22] use MDP and POMDP models with a fixed, predefined state space to represent the user's intentions. In contrast, the space of possible groundings is defined by objects and action available to the robot's perception; the user's intentions are defined by the grounding graph and vary according to the structure of the language.

TABLE II: Performance Deciding When to Ask a Question

	% Correct, oracle coreference	% Questions Asked
Metric 1 (Confidence)	71%	38%
Metric 2 (Entropy)	87%	40%

VI. 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. This system is able to ask the human user targeted questions about parts of a command that it 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, expanding the repertoire of questions and answers that the system can understand. Integrating nonverbal backchannels and gesture into the framework as new types of factors in the grounding graph remains an open problem. Finally, 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.

VII. Acknowledgements

This work was sponsored by the U.S Army Research Laboratory under the Robotics Collaborative Technology Alliance, by the Office of Naval Research under MURI N00014-07-1-0749, and by Battelle under Robert Carnes and Scott Sheaf. Their support is gratefully acknowledged. We would like to thank Javier Velez, Ashis Banerjee and Josh Joseph for helpful discussions about this work.

References

[1] Andrea Bauer, Klaas Klasing, Georgios Lidoris, Quirin Mhlbauer, Florian Rohrmller, Stefan Sosnowski, Tingting Xu, Kolja Khnlenz, Dirk Wollherr, and Martin Buss. The Autonomous City Explorer: Towards natural human-robot interaction in urban environments. *International Journal of Social Robotics*, 1(2):127–140, April 2009.

[2] Rehj Cantrell, Matthias Scheutz, Paul Schermerhorn, and Xuan Wu. Robust spoken instruction understanding for HRI. In *Proceeding of the 5th ACM/IEEE international conference on Human-robot interaction*, page 275282, 2010.

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

[4] J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn. 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 *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 4163–4168, 2009.

[5] Kai-yuh Hsiao, Stefanie Tellex, Soroush Vosoughi, Rony Kubat, and Deb Roy. Object schemas for grounding language in a responsive robot. *Connection Science*, 20(4):253–276, 2008.

[6] Ray S. Jackendoff. *Semantics and Cognition*, pages 161–187. MIT Press, 1983.

[7] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Pearson Prentice Hall, 2 edition, May 2008. ISBN 0131873210.

[8] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In *Proc. ACM/IEEE Int'l Conf. on Human-Robot Interaction (HRI)*, pages 259–266, 2010.

[9] Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proc. Nat'l Conf. on Artificial Intelligence (AAAI)*, pages 1475–1482, 2006.

[10] Cynthia Matuszek, Dieter Fox, and Karl Koscher. Following directions using statistical machine translation. In *Proc. ACM/IEEE Int'l Conf. on Human-Robot Interaction (HRI)*, pages 251–258, 2010.

[11] Stephanie Rosenthal, Manuela Veloso, and Anind K. Dey. Learning accuracy and availability of humans who help mobile robots. In *Proc. AAAI*, 2011.

[12] N. Roy, J. Pineau, and S. Thrun. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, 2000.

[13] Nobuyuki Shimizu and Andrew Haas. Learning to follow navigational route instructions. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, pages 1488–1493, 2009.

[14] Dimitar Simeonov, Stefanie Tellex, Thomas Kollar, and Nicholas Roy. Toward interpreting spatial language discourse with grounding graphs. In *2011 RSS Workshop on Grounding Human-Robot Dialog for Spatial Tasks*, 2011.

[15] M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, and D. Brock. Spatial language for human-robot dialogs. *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 34(2):154–167, 2004. ISSN 1094-6977.

[16] Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. Reconcile: A coreference resolution research platform. Technical report, Cornell University, April 2010.

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

[18] S. Tellex, T. Kollar, S. Dickerson, M.R. Walter, A. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proc. AAAI*, 2011.

[19] Adam Vogel and Dan Jurafsky. Learning to follow navigational directions. In *Proc. Association for Computational Linguistics (ACL)*, pages 806–814, 2010.

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

[21] Terry Winograd. *Procedures as a representation for data in a computer program for understanding natural language.* PhD thesis, Massachusetts Institute of Technology, 1970.

[22] S. Young. Using POMDPs for dialog management. In *IEEE Spoken Language Technology Workshop, 2006*, pages 8–13, 2006.