COMP 790.139 (Fall 2017) Natural Language Processing

Language+Robotics

Mohit Bansal
NLP for Actions/Robotics

- Task-based instructions, e.g., navigation, grasping, manipulation, skill learning

Turn right at the butterfly painting, then go to the end of the hall.
NLP for Actions/Robotics

- Task-based instructions, e.g., navigation, grasping, manipulation, skill learning

Cut some onions, and add to broth, stir it
Navigation Instruction Following

Abstract
The ability to understand natural-language instructions is critical to building intelligent agents that interact with humans. We present a system that learns to transform natural-language navigation instructions into executable formal plans. Given no prior linguistic knowledge, the system learns by simply observing how humans follow navigation instructions. The system is evaluated in three complex virtual indoor environments with numerous objects and landmarks. A previously collected realistic corpus of complex English navigation instructions for these environments is used for training and testing data. By using a learned lexicon to refine inferred plans and a supervised learner to induce a semantic parser, the system is able to automatically learn to correctly interpret a reasonable fraction of the complex instructions in this corpus.

1 Introduction
An important application of natural language processing is the interpretation of human instructions. The ability to parse instructions and perform the intended actions is essential for smooth interactions with a computer or a robot. Some recent work has explored how to map natural-language instructions into actions that can be performed by a computer (Branavan et al. 2009; Lau, Drews, and Nichols 2009). In particular, we focus on the task of navigation (MacMahon, Stankiewicz, and Kuipers 2006; Shimizu and Haas 2009; Matuszek, Fox, and Koscher 2010; Kollar et al. 2010; Vogel and Jurafsky 2010).

Figure 1: This is an example of a route in our virtual world. The world consists of interconnecting hallways with varying floor tiles and paintings on the wall (butterfly, fish, or Eiffel Tower.) Letters indicate objects (e.g. ‘C’ is a chair) at a location.
Navigation Instruction Following

Figure 2 shows our system's general framework. Given the sentences, all the actions are discrete and consist of turning left, turning right, and moving from one intersection to another. Since they did not have access to the overview map, they had to rely on observations for their explorations of the environments. These instructions were then given to several human followers whose actions were recorded as they tried to follow the instructions. On average, each instruction was 5 sentences long. However, to make the instruction generation process more manageable, we divided the sentences into smaller chunks and aligned them with their corresponding sentences to test the system's ability to understand and follow parts of the instructions. The human followers were then asked to give a set of written instructions on how to get from a particular position to another. Since they did not have access to the overview map, they had to rely on their explorations of the environments.

Instruction Following

- Observation
- World State
- Action Trace
- Instruction

Learning system for parsing navigation instructions
- Navigation Plan Constructor
- Plan Refinement
- Semantic Parser Learner
- Semantic Parser
- Execution Module (MARCO)

Training
- Instruction
- World State
- Action Trace

Testing
- Instruction
- World State
- Action Trace
We aim to design a semantic representation that is learnable, models grounded phenomena such as spatial language modeling. Our semantic representation combines ideas from meta-entities that specify actions in the world, (3) truth values that are objects in the world, (2) events, and example of semantics of spatial phrases. For example, the noun "chair" (Figure 2a) is paired with the expression \( \lambda x.\text{chair}(x) \).

\[
\begin{array}{c|cc|cc|c}
| & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \text{blue hall} & \text{blue hall} & \text{blue hall} & \text{blue hall} & \text{blue hall} \\
2 & \text{you} & \text{you} & \text{you} & \text{you} & \text{you} \\
3 & \text{chair} & \text{chair} & \text{chair} & \text{chair} & \text{chair} \\
4 & \text{hall} & \text{hall} & \text{hall} & \text{hall} & \text{hall} \\
5 & \text{inter} & \text{inter} & \text{inter} & \text{inter} & \text{inter} \\
\end{array}
\]

Figure 2: Schematic diagram of a map environment and example of semantics of spatial phrases.
Figure 4: A CCG parse showing adverbial phrases and topicalization.

Abstract

We propose a neural sequence-to-sequence model for direction following, a task that is essential to realizing effective autonomous agents. Our alignment-based encoder-decoder model with long short-term memory recurrent neural networks (LSTM-RNN) translates natural language instructions to action sequences based upon a representation of the observable world state. We introduce a multi-level aligner that empowers our model to focus on sentence “regions” salient to the current world state by using multiple abstractions of the input sentence. In contrast to existing methods, our model uses no specialized linguistic resources (e.g., parsers) or task-specific annotations (e.g., seed lexicons). It is therefore generalizable, yet still achieves the best results reported to-date on a benchmark single-sentence dataset and competitive results for the limited-training multi-sentence setting. We analyze our model through a series of ablations that elucidate the contributions of the primary components of our model.

Introduction

Robots must be able to understand and successfully execute natural language navigational instructions if they are to work seamlessly alongside people. For example, someone using a voice-commandable wheelchair might direct it to “Take me to the room across from the kitchen,” or a soldier may command a micro aerial vehicle to “Fly down the hallway into the second room on the right.” However, interpreting such free-form instructions (especially in unknown environments) is challenging due to their ambiguity and complexity, such as uncertainty in their interpretation (e.g., which hallway does the instruction refer to), long-term dependencies among both the instructions and the actions, differences in the amount of detail given, and the diverse ways in which the language can be composed. Figure 1 presents an example instruction that our method successfully follows.

Previous work in this domain (Chen and Mooney 2011; Chen 2012; Kim and Mooney 2012; 2013; Artzi and Zettlemoyer 2013; Artzi, Das, and Petrov 2014) largely requires specialized resources like semantic parsers, seed lexicons, and rerankers to interpret ambiguous, free-form natural language instructions. In contrast, the goal of our work is to learn to map instructions to actions in an end-to-end fashion that assumes no prior linguistic knowledge. Instead, our model learns the meaning of all the words, spatial relations, syntax, and compositional semantics from just the raw training sequence pairs, and learns to translate the free-form instructions to an executable action sequence.

We propose a recurrent neural network with long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) to both encode the navigational instruction sequence bidirectionally and to decode the representation to an action sequence, based on a representation of the current world state. LSTMs are well-suited to this task, as they have been shown to be effective in learning the temporal dependencies in natural language tasks.

Figure 1: An example of a route instruction-path pair in one of the virtual worlds from MacMahon, Stankiewicz, and Kuipers (2006) with colors that indicate floor patterns and wall paintings, and letters that indicate different objects. Our method successfully infers the correct path for this instruction.
After having explored an environment, objects (hat rack, lamp, chair, sofa, barstool, and easel) at intersections. Further, the model is only aware of the local environment. These investigational instructions to action sequences based only on the high-level input abstraction, but also the low-level representation of the input instruction, which improves performance.

The environments (Fig. 1) consist of interconnected virtual worlds. The environments (Fig. 1) consist of interconnected virtual environments in the agent's line-of-sight. We consider the problem of mapping natural language navigation instructions to action sequences, a technique that has proven effective in machine translation. However, unlike the standard alignment techniques, our model learns to align based not only on the correct action sequence but also the low-level representation of the input instruction, which improves performance.

The environment is a variable length natural language instruction, a technique that has proven effective in machine translation. However, unlike the standard alignment techniques, our model learns to align based not only on the correct action sequence but also the low-level representation of the input instruction, which improves performance.

Figure 2: Our encoder-aligner-decoder model with multi-level alignment


Figure 4: Visualization of the alignment between words to actions in a map for a multi-sentence instruction.
Navigation Instruction Following

Table 1: Overall accuracy (state-of-the-art in bold)

<table>
<thead>
<tr>
<th>Method</th>
<th>Single-sent</th>
<th>Multi-sent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chen and Mooney (2011)</td>
<td>54.40</td>
<td>16.18</td>
</tr>
<tr>
<td>Chen (2012)</td>
<td>57.28</td>
<td>19.18</td>
</tr>
<tr>
<td>Kim and Mooney (2012)</td>
<td>57.22</td>
<td>20.17</td>
</tr>
<tr>
<td>Kim and Mooney (2013)</td>
<td>62.81</td>
<td>26.57</td>
</tr>
<tr>
<td>Artzi and Zettlemoyer (2013)</td>
<td>65.28</td>
<td>31.93</td>
</tr>
<tr>
<td>Artzi, Das, and Petrov (2014)</td>
<td>64.36</td>
<td>35.44</td>
</tr>
<tr>
<td>Andreas and Klein (2015)</td>
<td>59.60</td>
<td>–</td>
</tr>
<tr>
<td>Our model (vDev)</td>
<td>69.98</td>
<td>26.07</td>
</tr>
<tr>
<td>Our model (vTest)</td>
<td><strong>71.05</strong></td>
<td>30.34</td>
</tr>
</tbody>
</table>
Navigation Instruction Following

Table 2: Model components ablations

<table>
<thead>
<tr>
<th></th>
<th>Full Model</th>
<th>High-level Aligner</th>
<th>No Aligner</th>
<th>Unidirectional</th>
<th>No Encoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-sentence</td>
<td>69.98</td>
<td>68.09</td>
<td>68.05</td>
<td>67.44</td>
<td>61.63</td>
</tr>
<tr>
<td>Multi-sentence</td>
<td>26.07</td>
<td>24.79</td>
<td>25.04</td>
<td>24.50</td>
<td>16.67</td>
</tr>
</tbody>
</table>

Table 3: Accuracy as a function of distance from destination

<table>
<thead>
<tr>
<th>Distance (d)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-sentence</td>
<td>71.73</td>
<td>86.62</td>
<td>92.86</td>
<td>95.74</td>
</tr>
<tr>
<td>Multi-sentence</td>
<td>26.07</td>
<td>42.88</td>
<td>59.54</td>
<td>72.08</td>
</tr>
</tbody>
</table>
The first three authors contributed equally to this paper.

Abstract

We frame the problem of following instructions as inferring plans given the environment and the command. In contrast, previous work in this area has focused on finding and executing plans corresponding to natural language commands such as “Put the tire pallet on the truck.” The hierarchical and compositional semantic structure introduced by Kollar et al. (2010) allows for the exploitation of this structure to infer the likelihood of a sequence of actions.

Our system performs inference in the model to successfully find and execute plans corresponding to natural language commands given to autonomous systems that perform navigation and mobile manipulation in semi-structured environments. Previous approaches have used models with fixed structure to infer the likelihood of a sequence of actions. However, this kind of fixed structure ignores the argument structure of natural language commands, in particular natural language command according to the compositional and hierarchical structure of a natural language command. Given a natural language command, the structure of the grounding graph model is induced using Spatial Description Clauses (SDCs), a semantically rich representation for grounding.

We model the grounding problem using a state space with a variable number of states and a variable number of actions, which can be performed in any order. The model is trained using a corpus of commands collected using crowdsourcing. We pair each command with robot actions such as “Get me the book,” “Put the tire pallet off the truck and set it down beside the truck,” “Get the tire pallet and place it on the pallet,” and “Pick up the pallet of boxes in the middle and place them on the trailer to the left.” We evaluate the robot’s performance by inferring plans from the corpus and use the corpus to learn the parameters of the model. We demonstrate that our system can successfully follow these commands.

Our system can successfully follow these commands. For example, “Get me the book” and “Put the tire pallet off the truck and set it down beside the truck,” has two arguments (“the box” and “on the truck”), each of which must be grounded to aspects of the world and which may be composed in many different ways. Figure 1 shows some of the wide variety of grounding actions that our system is able to follow and navigate.

In Proceedings of the National Conference on Artificial Intelligence (AAAI 2011).

1 Introduction

To be useful teammates to human partners, robots must be able to robustly follow spoken instructions. For example, to be useful teammates to human partners, robots must be able to successfully perform navigation and mobile manipulation in semi-structured environments. Previous approaches have used models with fixed structure to infer the likelihood of a sequence of actions. However, this kind of fixed structure ignores the argument structure of natural language commands, in particular natural language command according to the compositional and hierarchical structure of a natural language command. Given a natural language command, the structure of the grounding graph model is induced using Spatial Description Clauses (SDCs), a semantically rich representation for grounding.

We model the grounding problem using a state space with a variable number of states and a variable number of actions, which can be performed in any order. The model is trained using a corpus of commands collected using crowdsourcing. We pair each command with robot actions such as “Get me the book,” “Put the tire pallet off the truck and set it down beside the truck,” “Get the tire pallet and place it on the pallet,” and “Pick up the pallet of boxes in the middle and place them on the trailer to the left.” We evaluate the robot’s performance by inferring plans from the corpus and use the corpus to learn the parameters of the model. We demonstrate that our system can successfully follow these commands.

Figure 1: A target robotic platform for mobile manipulation and navigation (Teller et al., 2010), and sample commands from the domain, created by untrained human annotators. Our system can successfully follow these commands.
• factors. First we define the following random variables:
The graph has two types of nodes: random variables and

For a phrase such as “the pallet on the truck,”
the words of the relation field of the
SDC; if non-empty, always a child SDC.

Parser (de Marneffe, MacCartney, and Manning, 2006).
ford dependencies, which are extracted using the Stanford

Figure 2: (a) SDC tree for “Put the pallet on the truck.” (b) Induced graphical model and factorization.

Figure 3: (a) SDC tree for “Go to the pallet on the truck.” (b) A different induced factor graph from Figure 2. Structural differences between the two models are highlighted in gray.
Figure 4: A sequence of the actions that the forklift takes in response to the command, “Put the tire pallet on the truck.” (a) The search grounds objects and places in the world based on their initial positions. (b) The forklift executes the first action, picking up the pallet. (c) The forklift puts the pallet on the trailer.
Navigation + Manipulation Instructions

Fig. 1. An illustration of the robot trajectory $x(t)$ generated from planning constraints that were inferred from the natural language instruction “move near the red box and the blue crate” using the Distributed Correspondence Graph (DCG) model. The dark gray, light gray, and white regions represent the goal states, admissible states, and inadmissible states respectively. The variables $o_1 \ldots o_4$ identify the four objects in the environment model.
Navigation + Manipulation Instructions

Fig. 2. A parse tree for the sentence “move near the red box and the blue crate”. Part-of-speech tags in the parse tree are from the Penn Treebank [2].

Fig. 3. The factor graph resulting from the parse tree in Figure 2, used by the G^3 algorithm to infer the groundings of the instructions. Each linguistic is grounded to a object, location, or action through a factor that incorporates the grounding of its children. Black boxes, white spheres, and gray sphere are factors, known random variables, and unknown random variables respectively.
Navigation + Manipulation Instructions

(a) “go to the blue box”  
(b) “move towards the green object”  
(c) “travel to the orange object”

Fig. 7. Images of labeled trajectories generated by constraint and environment sampling that form the training and test sets for constraint inference evaluation.
### Configuration/Assembling Instructions

<table>
<thead>
<tr>
<th>Scene</th>
<th>Utterance</th>
<th>Error:</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Scene Image" /></td>
<td>Move the block that is currently located closest to the top left corner to the bottom left of the table, slightly higher than the block in the bottom right corner.</td>
<td>7.29 Block lengths</td>
</tr>
<tr>
<td><img src="image.png" alt="Error Image" /></td>
<td>Move the block closest to the top left corner so it is above half a block length to the right of the blocks near the lower left corner of the table.</td>
<td>0.94 Block lengths</td>
</tr>
</tbody>
</table>

**Table 6:** Above are two commands and the worlds they apply to. Below we see the prediction error of our best model.
Figure 2: Our models all follow the above architecture. 1-Hot word vectors (orange) are fed as input to a Feed-Forward or Recurrent Neural Network for encoding. A semantic representation is extracted (green), which in conjunction with knowledge of the world (blue) is grounded to predict an action.
Table 4: Model error when trained on only the subset of the data with decorated blocks or blank blocks. Where appropriate S, R, and D are the model’s predictive accuracy at identifying the Source, Reference and Direction. All models are evaluated on the Median and Mean prediction error the source block and its final target location. Distances are presented in block-lengths.
Move the block closest to the right table edge so it is to the left of the stack near the front left table corner.

Figure 1: An example of the configuration instruction understanding task (based on blank-labeled blocks). Our model is able to correctly predict the source block and the target position in this case.
Configuration/Assembling Instructions

![Diagram of the assembly instruction understanding task model](image)

Figure 2: Our overall model for the assembly instruction understanding task, showing instruction and world representation learning, language-to-block alignment modules, and source and target (expectation vs. sampling) loss functions.

## Configuration/Assembling Instructions

### Table 2: Final test results of our final sampling and expectation models (w/o and w/ ensemble), compared to the previous state-of-the-art on this dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>SOURCE</th>
<th>TARGET</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Median</td>
</tr>
<tr>
<td>End-to-End FFN (Bisk, Yuret, and Marcu 2016)</td>
<td>9.0%</td>
<td>3.45</td>
</tr>
<tr>
<td>End-to-End RNN (Bisk, Yuret, and Marcu 2016)</td>
<td>10.0%</td>
<td>3.29</td>
</tr>
<tr>
<td>Our Expectation Model</td>
<td>56.1%</td>
<td>0.00</td>
</tr>
<tr>
<td>Our Sampling Model</td>
<td>56.3%</td>
<td>0.00</td>
</tr>
<tr>
<td>Our Expectation Model w/ Ensemble</td>
<td>56.6%</td>
<td>0.00</td>
</tr>
<tr>
<td>Our Sampling Model w/ Ensemble</td>
<td>56.8%</td>
<td>0.00</td>
</tr>
</tbody>
</table>

The box in the bottom right, slightly right of center, moves one space north of the tower.

Move the highest block down to below and in front of the right stack of blocks.

Take the leftmost front block and place it on top of the stack of two blocks furthest to the back.

Take the block from the last row and hide it behind the tower.

Positive Examples
Configuration/Assembling Instructions

Move the block closest to the bottom left corner so that it is on top of the block at the top of the backwards L.

The box next to the Tetris structure moves two spaces left and one half up.

Slide the block left of the two in the top right over and on top of the block in front of the tower.

In the 3-piece-long line, the middle box takes a second story from the middle box in the top row.

Negative Examples

Fig. 2 The human interaction with the BakeBot system for recipe execution. First the person provides the plain-text recipe and the measured ingredients. Then BakeBot infers a sequence of baking primitives to execute that correspond to following the recipe. If BakeBot encounters an unsupported baking primitive, it asks its human partner for help executing the instruction. The end result is baked cookies.
Recipe Instruction Following

Fig. 3 Architecture of the BakeBot system. The NL system processes the plain text recipe, producing a high-level plan which is sent to the robot. For each instruction in the high-level plan, the motion planner assembles a motion plan and executes it on the PR2 robot.

http://projects.csail.mit.edu/video/research/robo/bakebot_final.mp4
Recipe Instruction Following

<table>
<thead>
<tr>
<th>Recipe Text</th>
<th>Inferred Action Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afghan Biscuits</td>
<td></td>
</tr>
<tr>
<td>200g (7 oz) butter</td>
<td>pour(butter, bowl); mix(bowl)</td>
</tr>
<tr>
<td>75g (3 oz) sugar</td>
<td>pour(sugar, bowl); mix(bowl)</td>
</tr>
<tr>
<td>175g (6 oz) flour</td>
<td>pour(flour, bowl); pour(cocoa, bowl)</td>
</tr>
<tr>
<td>25g (1 oz) cocoa powder</td>
<td>pour(cornflakes, bowl); mix(bowl)</td>
</tr>
<tr>
<td>50g cornflakes (or crushed weetbix)</td>
<td>scrape()</td>
</tr>
<tr>
<td>Soften butter.</td>
<td>preheat(350); bake(pan, 20)</td>
</tr>
<tr>
<td>Add sugar and beat to a cream.</td>
<td></td>
</tr>
<tr>
<td>Add flour and cocoa.</td>
<td></td>
</tr>
<tr>
<td>Add cornflakes last.</td>
<td></td>
</tr>
<tr>
<td>Put spoonfuls on a greased oven tray.</td>
<td></td>
</tr>
<tr>
<td>Bake about 15 minutes at 180°C (350°F).</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4 Text from a recipe in our dataset, paired with the inferred action sequence for the robot.
Recipe Instruction Following

Recipe Instruction Following

Recipes: Tell Me Dave (http://tellmedave.cs.cornell.edu/)

Fig. 1. Natural Language Instructions to sequence of instructions for a given new environment. Our approach takes description in natural language and sequences together robotic instructions that are appropriate for a given environment and task. Note that the NL instructions are often ambiguous, and are incomplete, and need to be grounded into the environment.
Recipe Instruction Following

- Recipes: Tell Me Dave (http://tellmedave.cs.cornell.edu/)

Fig. 4. **Robot Experiment.** Given the language instruction for making the dessert ‘Affogato’: ‘Take some coffee in a cup. Add ice cream of your choice. Finally, add raspberry syrup to the mixture.’, our algorithm outputs a sequence that the PR2 executes to make the dessert. (Please see the video.)
Fig. 5. **Screen-shot of Robobarista**, the crowd-sourcing platform running on Chrome browser. We have built Robobarista platform for collecting a large number of crowd demonstrations for teaching the robot.
Recipe Instruction Following


---

**Fig. 10.** Examples of transferred trajectories being executed on PR2. On the left, PR2 is able to rotate the ‘knob’ to turn the lamp on. In the third snapshot, using two transferred trajectories, PR2 is able to hold the cup below the ‘nozzle’ and press the ‘lever’ of ‘coffee dispenser’. In the last example, PR2 is frothing milk by pulling down on the lever, and is able to prepare a cup of latte with many transferred trajectories.
Navigation Instruction Generation

Input: map and path

Floor patterns:
- Blue
- Brick
- Concrete
- Grass
- Black
- Wood
- Yellow

Wall paintings:
- Tower
- Butterfly
- Fish

Objects:
- B Barstool
- C Chair
- E Easel
- H Hatrack
- L Lamp
- S Sofa

Output: route instruction

“turn to face the grass hallway. walk forward twice. face the easel. move until you see black floor to your right. face the stool. move to the stool”

Fig. 4. Participants’ field of view in the virtual world used for the human navigation experiments.

Fig. 1. An example route instruction that our framework generates for the shown map and path.
Navigation Instruction Generation

Fig. 2. Our method generates natural language instructions for a given map and path.

Fig. 3. Our encoder-aligner-decoder model for surface realization.
Alignment visualization for two pairs of CAS (left) and natural language instructions (top). Darker colors denote greater attention weights.

Fig. 5.

Map and Paths

Legend:
- H - Hatrack
- B - Barstool
- C - Chair
- S - Sofa
- L - Lamp
- Fish
- Eiffel
- Butterfly

Fig. 8. Examples of paths from the SAIL corpus that ten participants (five for each map) followed according to instructions generated by humans and by our method. Paths in red are those traversed according to human-generated instructions, while paths in green were executed according to our instructions. Circles with an “S” and “G” denote the start and goal locations, respectively.
Language Generation/Dialogue by Robots

Navigation Dialogue

Fig. 1. A user gives a tour to a robotic wheelchair designed to assist residents in a long-term care facility. (Left) The guide provides an ambiguous description of the kitchen’s location. (Right) When the robot is near one of the likely locations, it asks the guide a question to resolve the ambiguity.
Language Generation/Dialogue by Robots

Manipulation Dialogue

(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 $\gamma_2$, $\gamma_3$ and $\gamma_5$ 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.
Manipulation Dialogue

3. Technical Approach

When faced with a command, the system parses the language into corresponding grounding graphs and performs inference to find the most likely set of values for the grounding variables $\gamma_1 \ldots \gamma_N$. The results described in this paper use ground-truth parses, but automatic parsing strategies are also possible.

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 the 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.

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.
Manipulation Dialogue

**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.
Learning New Actions via Dialogue

Figure 1: An example setup and dialogue. Objects are marked with labels only for the illustration purpose.
and executability of actions, and representing and reasoning with the preconditions, effects, et al. symbolic planner formalized using action language typing in natural language. The agent generated queries to interacted with the agent through a graphical user interface by 6.1 Implementation (Segbot) as shown in Figure 5 (Left) using the Robot Operator.

The agent was integrated into a Segway-based robot platform 6 Segbot Experiments. In addition to learning misspelling corrections and new referring expressions, the agent learned to parse things like "item in calender" and "item in planner" from user goals. Action languages are used for logical facts provided to the symbolic planner. For instance, we used laser sensors to detect whether office doors were logical facts provided to the symbolic planner. For instance, the actions taken were correct. If they answered "yes" and the robot platform (Segbot) used in experiments had access to only the seed lexicon. Because of this conversation, the agent learned that "calender" and "planer" mean "calendar" during retraining.

Figure 3: The Mechanical Turk interface for the delivery task. This abridged conversation is from a Turker in training batch 0, when the system had access to only the seed lexicon. Because of this conversation, the agent learned that "calender" and "planner" mean "calendar" during retraining.

During training, the robot understood and carried out 35 conversations the agent had over a few days' time. We intended to replace our static dialog policy with a reinforcement learning based on parser confidence and conversation success. We will also explore whether our approach can be extended to include dialog management. We also plan to incorporate semantic parsing and dialog management. We thank the anonymous reviewers for their feedback. A portion of this work has taken place in the Learning Agents Research Group.

We note that there is significant improvement in user performance in the former set as compared to the latter set as seen once and each navigation test goal was seen 5 times). We then allowed the agent to perform incremental learning over the whole phrase to this meaning. Means in underlined bold differ significantly (p<0.05) from user goals. For testing, users were given one goal from the navigation and delivery tasks. Users in test conditions did not significantly favor using the robot for tasks after conversation length, as a metric for dialog efficiency. For instance, the latter set as seen once and each navigation test goal was seen 5 times). We evaluated our agent's initial performance by giving 10 users one of each of these goals (so each delivery test goal was included the directory panels used in the Mechanical Turk experiments with hundreds of users. A proof-

We then allowed the agent to perform incremental learning over the whole phrase to this meaning. Means in underlined bold differ significantly (p<0.05) from user goals. For testing, users were given one goal from the navigation and delivery tasks. Users in test conditions did not significantly favor using the robot for tasks after conversation length, as a metric for dialog efficiency. For instance, the latter set as seen once and each navigation test goal was seen 5 times).

The agent was integrated into a Segway-based robot platform (Segbot) as shown in Figure 5 (Left) using the Robot Operator.

Figure 5: Left: Robot platform (Segbot) used in experiments. Right: Segbot architecture, implemented using Robot Operating System (ROS).

Table 2 compares the survey responses of users and the number of goals users completed of each task type in the segmentation of user survey responses. The number of goals users completed of each task type in the survey.

6.2 Methodology

Figure 5: Left: Robot platform (Segbot) used in experiments. Right: Segbot architecture, implemented using Robot Operating System (ROS).