Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011) (2011), pp. 859-865.

Learning to Interpret Natural Language Navigation Instructions from Observations

David L. Chen and **Raymond J. Mooney**
Department of Computer Science
The University of Texas at Austin
1616 Guadalupe, Suite 2.408
Austin, TX 78701, USA
dlcc@cs.utexas.edu and mooney@cs.utexas.edu

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).

The goal of the navigation task is to take a set of natural-language directions, transform it into a navigation plan that can be understood by the computer, and then execute that plan to reach the desired destination. Route direction is a unique form of instructions that specifies how to get from one place to another and understanding them depends heavily on the spatial context. The earliest work on interpreting route directions was by linguists (Klein 1982; Wunderlich and Reinelt 1982). While this domain is restricted, there is considerable variation in how different people describe the same route. Below are some examples from our test corpus of instructions given for the route shown in Figure 1:

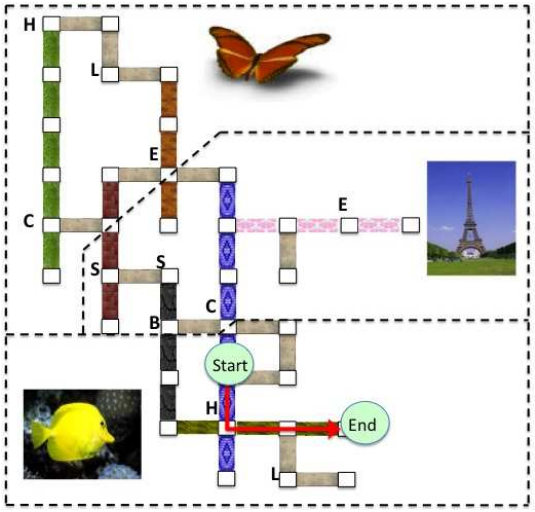

Figure 1: This is an example of a route in our virtual world. The world consists of interconnecting hallways with varying floor tiles and paintings on the wall (butterfly, fish, or Eiffel Tower.) Letters indicate objects (e.g. 'C' is a chair) at a location.

> "Go towards the coat rack and take a left at the coat rack. go all the way to the end of the hall and this is 4."
>
> "Position 4 is a dead end of the yellow floored hall with fish on the walls."
>
> "turn so that the wall is on your right side. walk forward once. turn left. walk forward twice."
>
> "foward to the fish. first left. go tot [sic] the end."
>
> "Place your back to the wall of the 'T' intersection. Turn right. Go forward one segment to the intersection with the yellow-tiled hall. This intersection contains a hatrack. Turn left. Go forward two segments to the end of the hall. This is Position 4."

As seen in these examples, people may describe routes using landmarks (e.g. *yellow floored hall*) or specific actions (e.g. *walk forward once*). They may describe the same object differently (e.g. *coat rack* vs. *hatrack*). They also differ in the amount of detail given, from just information about the

destination (e.g. *Position 4 is a dead end ...*) to step-by-step instructions along with verification steps (e.g. *This intersection contains a hatrack*). Thus, even ignoring spelling and grammatical errors as well as logical errors (e.g. confusing left and right), navigation instructions can be quite diverse and contain different information which makes interpreting them a challenging problem.

In this paper we introduce a general framework for learning to interpret navigation instructions given only sample observations of humans following such instructions. The system first infers a formal navigation plan for each instruction based on the observed actions. Using this as supervision, it then learns a semantic parser that can map novel instructions into navigation plans executable by a (simulated) robot. Using a learned lexicon to refine the plans is shown to help correctly infer more complex navigation plans. This is vital in successfully following long instructions where error recovery is necessary.

The rest of the paper is organized as follows. We first review relevant work in the area of learning natural language instructions and grounded language acquisition in Section 2. We then formally define our learning problem and the virtual environment we use to test our navigation system in Section 3. Details of our system are described in Section 4. We present experimental results in Section 5. Finally, we discuss possible future work in Section 6 and conclude in Section 7.

2 Related Work

Building systems that learn to interpret navigation instructions has recently received some attention due to its application in building mobile robots. Our work is the most similar to that of Matuszek et al. (2010). Their system learns to follow navigation instructions from example pairs of instructions and map traces with no prior linguistic knowledge. They used a general-purpose semantic parser learner WASP (Wong and Mooney 2006) to learn a semantic parser and constrain the parsing results with physical limitations imposed by the environment. However, their virtual world is relatively simple with no objects or attribute information as it is constructed from laser sensors.

Similarly, Shimizu and Haas (2009) built a system that learns to parse navigation instructions. They restrict the space of possible actions to 15 labels and treat the parsing problem as a sequence labeling problem. This has the advantage that context of the surrounding instructions are taken into account. However, their formal language is very limited in that there are only 15 possible parses for an instruction.

There is some recent work that explores direction following in more complex environments. Vogel and Jurafsky (2010) built a learning system for the HCRC Map Task corpus (Anderson et al. 1991) that uses reinforcement learning to learn to navigate from one landmark to another. The environment consists of named locations laid out on a map. Kollar et al. (2010) presented a system that solves the navigation problem for a real office environment. They use LIDAR and camera data collected from a robot to build a semantic map of the world and to simulate navigation. However, both of these systems were directly given object names or required other resources to learn to identify objects in the world. Moreover, both systems used lists of predefined spatial terms. In contrast, we do not assume any existing linguistic knowledge or resource.

Besides navigation instructions, there has also been work on learning to interpret other kinds of instructions. Recently, there has been some interest in learning how to interpret English instructions describing how to use a particular website or perform other computer tasks (Branavan et al. 2009; Lau, Drews, and Nichols 2009). These systems learn to predict the correct computer action (pressing a button, choosing a menu item, typing into a text field, etc.) corresponding to each step in the instructions.

Our work also fits into the broader area of *grounded language acquisition*, in which language is learned by simply observing its use in some naturally occurring perceptual context (see Mooney (2008) for a review). Unlike most work in statistical NLP which requires annotating large corpora with detailed syntactic and/or semantic markup, this approach tries to learn language without explicit supervision in a manner more analogous to how children acquire language. This approach also grounds the meaning of words and sentences in perception and action instead of arbitrary semantic tokens. One of the core issues in grounded language acquisition is solving the correspondence between language and the semantic context. Various approaches have been used including supervised training (Snyder and Barzilay 2007), iteratively retraining a semantic parser/language generator to disambiguate the context (Kate and Mooney 2007; Chen, Kim, and Mooney 2010), building a generative model of the content selection process (Liang, Jordan, and Klein 2009; Kim and Mooney 2010), and using a ranking approach (Bordes, Usunier, and Weston 2010). Our work differs from these previous approaches in that we explicitly model the relationships between the semantic entities rather than treating them as individual items.

3 Problem Definition and Evaluation Data

The goal of the navigation problem is to build a system that can understand free-form natural-language instructions and follow them to move to the desired destination. In particular, we approach the problem assuming no prior linguistic knowledge: syntactic, semantic, or lexical. This means we have to learn the meaning of every word, including object names, verbs, spatial relations, as well as the syntax and compositional semantics of the language. The only supervision we receive is in the form of observing how humans behave when following sample navigation instructions.

Formally, the system is given training data in the form: $\{(e_1, a_1, w_1), (e_2, a_2, w_2), \ldots, (e_n, a_n, w_n)\}$, where e_i is a natural language instruction, a_i is an observed action sequence, and w_i is a description of the current state of the world including the patterns of the floors and walls and positions of any objects. The goal is then to build a system that can produce the correct a_j given a previously unseen (e_j, w_j) pair.

The main challenge of this problem is that the navigation plans described by the instructions are not directly observed. As the example in Section 1 showed, several different plans can be used to navigate the same route. In other words, there

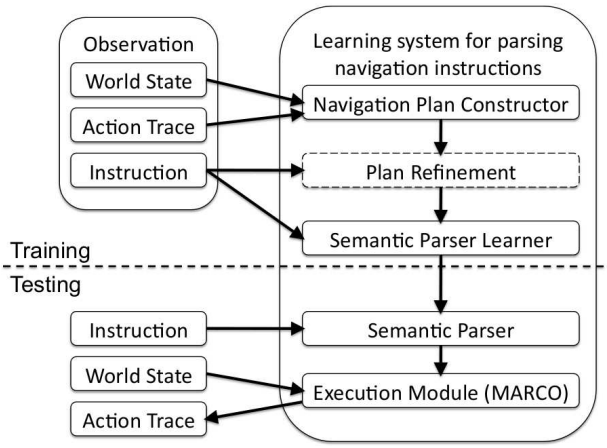

Figure 2: An overview of our system

is not always a direct correspondence between e_i and a_i. Rather, e_i corresponds to an unobserved plan p_i that when executed in w_i will produce a_i. Thus, we need to first infer the correct p_i from the training data and then build a semantic parser that can translate from e_i to p_i.

To train and test our system, we use the data and virtual environments assembled by MacMahon *et al.* (2006). The data was collected for three different virtual worlds consisting of interconnecting hallways. An overview map of one of the worlds is shown in Figure 1. Each world consists of several short concrete hallways and seven long hallways each with a different floor pattern (grass, brick, wood, gravel, blue, flower, and yellow octagons). The worlds are divided into three areas, each with a different painting on the walls (butterfly, fish, and Eiffel Tower). There is also furniture placed at various intersections (hatrack, lamp, chair, sofa, barstool, and easel). The three worlds contain the same elements but in different configurations. Each world also has seven chosen positions labeled 1 thorough 7.

MacMahon *et al*. collected both human instructor data and human follower data. The instructors first familiarized themselves with the environment and the seven positions. They 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. 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 simplify the learning problem, we manually split the action sequences and aligned them with their corresponding sentences. All the actions are discrete and consist of turning left, turning right, and moving from one intersection to another.

4 System Description

Figure 2 shows our system's general framework. Given the observation (w_i, a_i, e_i), we first construct a formal navigation plan p_i based on the action sequence a_i and the world state w_i. An optional step refines this navigation plan based on the instruction e_i. The resulting pair (e_i, p_i) is then used as supervised training data for learning a semantic parser. During testing, the semantic parser maps new instructions e_j into formal navigation plans p_j which are then carried out by the execution module.

Instruction: "Go away from the lamp to the intersection of the red brick and wood"

Basic:
Turn ( ) ,
Travel (steps: 1)

Landmarks:
Turn ( ) ,
Verify (left: WALL , back: LAMP , back: HATRACK , front: BRICK HALL) ,
Travel (steps: 1) ,
Verify (side: WOOD HALL)

Figure 3: Examples of automatically generated plans.

While we built the top two components that are responsible for creating the supervised training data (e_i, p_i), we use existing systems for building semantic parsers and for executing navigation plans. Since the plans inferred by the system are not always completely accurate representations of the instructions, we chose a semantic-parser learner, KRISP, that has been shown to be particularly robust to noisy training data (Kate and Mooney 2006). Nevertheless, other general-purpose supervised semantic-parser learners (Zettlemoyer and Collins 2005; Wong and Mooney 2006; Lu et al. 2008) could also be used. To carry out the plans, we use MARCO's execution module developed by MacMahon *et al.* (2006) for executing navigation plans in our test environments.

4.1 Constructing navigation plans

A simple way to generate a formal navigation plan is to model only the observed actions. In our case, this means forming plans that consist of only turning left and right, and walking forward a certain number of steps. This is often sufficient if the instruction directly refers to the specific action to be taken (e.g. *turn left*, *walk forward two steps*). We refer to these navigation plans which capture such direct instructions as *basic plans*.

To capture more complex instructions that refer to objects and locations in the environment (e.g. *face the pink flower hallway*, *go to the sofa*), we simulate executing the given actions in the environment. We collect sensory data during the execution and form a *landmarks plan* that adds interleaving verification steps to the *basic plan*. The verification steps specify the landmarks that should be observed after executing each basic action. Examples of both a *basic plan* and a *landmarks plan* are shown in Figure 3.

4.2 Plan refinement

While *landmarks plans* capture more of the meaning of the instructions, they can also contain a lot of extraneous information. Thus, we employ a lexicon learning algorithm to learn the meanings of words and short phrases. The learned lexicon is then used to try to identify and remove the extraneous details in the *landmarks plan*.

Learning a lexicon We build a semantic lexicon by finding the common parts of the formal representations associ-

Algorithm 1 LEXICON LEARNING

input Navigation instructions and the corresponding navigation plans $(e_1, p_1), \ldots, (e_n, p_n)$
output $Lexicon$, a set of phrase-meaning pairs

main
  for n-gram w that appears in $\mathbf{e} = (e_1, \ldots, e_n)$ **do**
    for instruction e_i that contains w **do**
      Add navigation plan p_i to $meanings(w)$
    end for
    repeat
      for every pair of meanings in $meanings(w)$ **do**
        Add intersections of the pair to $meanings(w)$
      end for
      Keep k highest-scoring entries of $meanings(w)$
    until $meanings(w)$ converges
    Add entries of $meanings(w)$ with scores higher than threshold t to $Lexicon$
  end for
end main

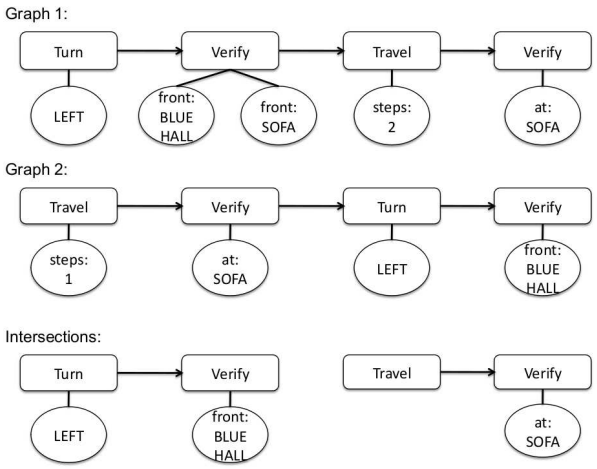

Figure 4: Example of computing the intersections of two graph representations of navigation plans.

ated with different occurrences of the same word or phrases (Siskind 1996). More specifically, we represent the navigation plans in graphical form and compute common parts by taking intersections of the two graphs (Thompson and Mooney 2003). Pseudo-code for the approach is shown in Algorithm 1. Initially, all navigation plans whose instruction contains a particular n-gram w are added to $meanings(w)$, the set of potential meanings of w. Then, the algorithm repeatedly computes the intersections of all pairs of potential meanings and adds them to $meanings(w)$ until further intersections do not produce any new entries. The intersection operation is performed by greedily removing the largest common subgraph from both graphs until the two graphs have no overlapping nodes. The output of the intersection process consists of all the removed subgraphs. An example of the intersection operation is shown in Figure 4. Each potential word-meaning pair is given a *score* (described below) that evaluates its quality. After $meanings(w)$ converges, its members with scores higher than a given threshold are added as lexical entries for w. In all our experiments, we consider only unigrams and bigrams, and use threshold $t = 0.4$ and maximum meaning set size $k = 1000$.

We use the following scoring function to evaluate a pair of an n-gram w and a graph g:

$$Score(w, g) = p(g|w) - p(g|\neg w)$$

Intuitively, the score measures how much more likely a graph g appears when w is present compared to when it is not. A good (w, g) pair means that w should be indicative of g appearing (i.e. $p(g|w)$ should be close to 1), assuming w is monosemous[1]. However, the reverse is not true since an object or action may often be referred to using other expressions or omitted from an instruction altogether. Thus, the absence of a word w when g occurs, $p(\neg w|g)$, is not evidence against g being the meaning of w. To penalize g's that are ubiquitous, we subtract the probability of g occurring when w is not present. We estimate all the probability measures by counting how many examples contain the words or the graphs, ignoring multiple occurrences in a single example.

[1]Notice that the actual magnitude of $p(g|w)$ matters. Thus, using odds ratios as the scoring function did not work as well.

Refining navigation plans using the lexicon The learned lexicon is then used to remove extraneous components from the *landmarks plans*. Ideally, a refined plan only contains actions and objects referred to in the instructions. However, we want to be conservative in pruning nodes so that important information is not removed from the data given to the semantic-parser learner. Therefore, nodes are only removed if it is quite certain they are not mentioned in the instruction since they do not correspond to the meaning of some word in the instruction.

To refine (e_i, p_i), we first select the highest-scoring lexical entry (w, g) such that w and g appear in e_i and p_i, respectively. We then remove w from e_i and mark all occurrences of g in p_i, ignoring any redundant markings. This process is repeated until no words remain in e_i or no more lexical entries can be selected. Finally, we remove all nodes in p_i that were not marked and the remaining graph becomes the new *refined plan* p_i'.

4.3 Learning a semantic parser

Once we obtain the supervised data in the form of (e_i, p_i), we use KRISP (Kate and Mooney 2006) to learn a semantic parser that can transform novel instructions e_j into navigation plans p_j (i.e. transform *turn to face the sofa* into Turn(), Verify(front: SOFA).)

KRISP is a publicly available learning algorithm for training parsers that translate from natural language strings to a formal language defined by a context-free grammar (CFG). Given parallel training data in the form of natural language strings with their corresponding formal meaning representations, it learns a set of string classifiers that decide how to construct meaning representations. In particular, it uses support vector machines (SVM) classifiers with string ker-

	Original	Single-sentence
# instructions	706	3236
Vocabulary size	660	629
Avg. # sentences	5.0 (2.8)	1.0 (0)
Avg. # words	37.6 (21.1)	7.8 (5.1)
Avg. # actions	10.4 (5.7)	2.1 (2.4)

Table 1: Statistics about the original corpus collected by MacMahon et al. as well as the segmented version of it that we use for learning. The average statistics for each instruction are shown with standard deviations in parentheses.

nels to decide when a word or phrase is indicative of a production rule of the CFG being applied. When semantically parsing a sentence, each classifier estimates the probability of each production rule covering different substrings of the sentence. This information is then used to compositionally build a complete string in the formal language for the sentence. Given the partial matching provided by string kernels and the over-fitting prevention provided by SVMs, KRISP has been shown to be robust to noisy training data (Kate and Mooney 2006).

4.4 Executing instructions

After semantically parsing the instruction, we need to execute the navigation plan to reach the intended destination. We use the execution module in MARCO (MacMahon, Stankiewicz, and Kuipers 2006) for this purpose. MARCO is a system that is designed to follow free-form natural language route instructions in our test environment. Using a syntactic parser and hand-engineered transformation rules for encoding knowledge about object names, verbs and spatial relationships, raw text is first translated into a *compound action specification*. The executor then carries out the specification by interleaving actions and perceptions to gain knowledge of the environment and to execute the actions. It has error recovery mechanisms for reconciling conflicting specifications (e.g. if the instruction is *walk two steps to the chair* when the chair is actually three steps away) and for inferring implicit commands.

To execute a navigation plan, we first transform it from our formal representation into a *compound action specification* and then use MARCO's executor to carry out the actions in the virtual worlds.

5 Experimental Evaluation

To evaluate our approach, we use the instructions and follower data collected by MacMahon *et al.* (2006) to train and test our system. The data contains 706 non-trivial route instructions for three virtual worlds. The instructions were produced by six instructors for 126 unique starting and ending position pairs spread evenly across the three worlds. There were 1 to 15 human followers for each instruction.

Since this data was originally collected only for testing purposes and not for learning, each instruction is quite long with an average of 5 sentences. However, for learning, it is more natural to observe the instructors interact with the followers as they progress. Thus, to create our training data, we first segmented the instructions into individual sentences. Then for each sentence, we paired it with an action sequence based on the majority of the followers' actions and our knowledge of the map. During this process, close to 300 sentences that could not be matched to any actions were discarded. Most of them were of the form "This is position n". Statistics for the original and segmented data can be seen in Table 1. We use the single-sentence version of the corpus for training and both versions for testing.

	Precision	Recall	F1
Basic plans	81.46	55.88	66.27
Landmarks plans	45.42	85.46	59.29
Refined landmarks plans	78.54	78.10	78.32

Table 2: Partial parse accuracy of how well the inferred navigation plans match the human annotations.

5.1 Generating navigation plans

We first examine how well our system infers the correct navigation plans from the observed actions. To do this, we hand-annotated each instruction in the single-sentence corpus with the correct navigation plans and compared the inferred plans to these gold-standard plans. We used a partial correctness metric to measure the precision and recall of the inferred plans. To calculate precision, each step in the inferred plan receives one point if it matches the type of the corresponding step in the gold-standard plan. An additional point is then awarded for each matching argument. Precision is computed as the sum of the points divided by the total number of possible points. Since the two plans may contain different number of steps, we used a dynamic programming algorithm to find a order-preserving mapping of steps from one plan to the other such that precision is maximized. Recall is computed similarly with the roles of the inferred and gold-standard plans swapped. We also compute F1 score, the harmonic mean of precision and recall.

The results are shown in Table 2. Since the *basic* and *landmarks plans* do not require training, their results are simply the average accuracy of the generated plans for all the examples. For the *refined landmarks plans*, the lexicon is trained on examples from two of the three maps and used to refine plans from the same two maps. The results are averaged over the three experiments. Compared to the *basic plans*, *landmarks plans* have better recall but considerably lower precision. However, if we use the lexicon to help refine these plans then we retain both the high precision of the *basic plans* and the high recall of the *landmarks plans*. This indicates the system is inferring fairly accurate plans which in turn produces reasonably accurate supervised examples for training the semantic parser.

5.2 Building a semantic parser

Next, we evaluated the performance of the semantic parsers trained on these inferred plans as well as semantic parsers trained on human-annotated plans. We used a leave-one-

	Precision	Recall	F1
Basic plans	86.68	48.62	62.21
Landmarks plans	50.40	31.10	38.39
Refined landmarks plans	90.22	55.10	68.37
Human annotated plans	88.24	71.70	79.11

Table 3: Partial parse accuracy of how well the semantic parsers trained on the different navigation plans performed on test data.

	Single-sentence	Complete
Simple generative model	11.08	2.15
Basic plans	56.99	13.99
Landmarks plans	21.95	2.66
Refined landmarks plans	54.40	16.18
Human annotated plans	58.29	26.15
MARCO	77.87	55.69
Human followers	N/A	69.64

Table 4: Experimental results comparing different versions of our system and several baselines on following both the single-sentence and complete instructions. The numbers are the percentages of trials reaching the correct destinations.

map-out approach where the semantic parser is trained on examples from two maps and tested on instructions from the third, unseen map. The parse outputs are compared to human annotations using partial correctness as before. The results are shown in Table 3. As expected, semantic parsers trained with cleaner data performed better. However, one thing to note is that precision of the training data is more important than recall. In particular, semantic parsers trained on *landmark plans* performed the worst in all aspects despite the plans having relatively high recall. This suggests the amount of noise exceeded what could be handled by KRISP and the system fails to learn to generalize properly. Thus, our refinement step is vital in keeping the plans relatively clean in order for KRISP to learn effectively.

5.3 Executing navigation plans

Next, we tested our end-to-end system by executing the parsed navigation plans to see if they reach the desired destinations. We evaluated on both the single-sentence and complete (multi-sentence) versions of the corpus. We employ a strict metric in which a trial is successful if and only if the final position (and orientation for the single-sentence version) exactly matches the intended position. This makes the experiments on the complete instructions especially difficult since *any* error parsing *any* of the sentences in the instruction can lead to a failure on the task. We again performed leave-one-map-out cross-validation. For each plan, we executed it 10 times since the execution component is non-deterministic when the plan is underspecified (e.g. the plan specifies a turn, but does not specify any directions or post-conditions). The average results are shown in Table 4.

In addition to evaluating the trained semantic parsers, we also compare to several other lower and upper baselines. We

Instruction: "Place your back against the wall of the 'T' intersection. Turn left. Go forward along the pink-flowered carpet hall two segments to the intersection with the brick hall. This intersection contains a hatrack. Turn left. Go forward three segments to an intersection with a bare concrete hall, passing a lamp. This is Position 5."

Parse:
Turn ( ),
Verify (back: WALL),
Turn (LEFT),
Travel ( ),
Verify (side: BRICK HALLWAY),
Turn (LEFT),
Travel (steps: 3),
Verify (side: CONCRETE HALLWAY)

Figure 5: A plan produced by the semantic parser trained on *refined landmarks plans*. While the parser misses some of the redundant information, the plan contains sufficient details to lead to the correct destination.

constructed a lower baseline that does not utilize any of the linguistic information in the instructions. Instead, it builds a simple generative model of the actions in the training data. During testing, the generative model first selects the number of actions to perform for each instruction, and then stochastically generates the action type and arguments. The low performance of this baseline indicates that the task is non-trivial even though there are only few available actions (turning and walking forward).

For the upper baselines, we compared to three different performances. First, we compare the performance of the semantic parser trained on human-annotated plans as before. This represents what could be achieved if we reconstructed the navigations plans in the training examples perfectly. Both the *basic plans* and *refined landmarks plans* approach this performance on the simpler, single-sentence task. To better understand what could be achieved in an engineered (non-learning) system, we also compared to the full MARCO system that parses and executes instructions. Finally, we also compared to the performance of human followers who tried to follow these instructions. While none of our systems perform as well as MARCO, it is important to note that our system must learn the complete language interpreter just from observations. Moreover, our system could be easily adapted to other languages and environments with different objects and landmarks. On the other hand, MARCO was fully manually-engineered for this environment and hand-tuned on this data to achieve the best performance. As expected, human followers performed the best, although even they were only able to complete 70% of the tasks [2], indicating the difficulty of the complete task.

Of the different versions of our system, *landmarks plans* performed the worst as expected because it failed to learn an effective semantic parser. The systems trained on *basic plans* and *refined landmarks plans* both perform similarly on this final execution task, with *basic plans* performing slightly better on the single-sentence task and *refined landmarks*

[2] Sometimes the instructions were wrong to begin with, since they were recreated from memory by the instructors.

plans performing better on the complete task. The better performance of the *basic plans* on the single-sentences task shows that for these shorter instructions, directly modeling the low-level actions is often sufficient. The additional benefit of modeling landmarks is not seen until testing on complete instructions. In this case, landmarks are often vital for recovering from small mistakes in the instructions or the parsing, or both. The system using *refined landmarks plans* performed the best out of the three variations in this setting, matching the trend observed in the parse-accuracy experiments (Table 3). A sample parse for this system is shown in Figure 5. While the plan is not a perfect representation of the instruction, it contains sufficient details to complete the task successfully in all trials.

6 Future Work

Currently, our system goes through the various stages of learning in a pipelined manner. Consequently, a mistake made in earlier steps will propagate to later stages. A better approach would be to build feedback loops to iteratively improve the estimates in each stage. Moreover, since these are executable actions, we can test our understanding of the language in the environment itself to receive additional reinforcements. However, it should be remembered that reaching the correct destination is not necessary indicative of a good plan (e.g. *landmarks plans* always lead to the correct destinations but do not correspond to the actual instructions.)

Semantic parsing is only half of what is required for language acquisition. The other half is language generation. Since our work is focused on resolving referential ambiguity, there is no inherent limitation to extending our approach to language generation as well. Similar to Chen *et al*. (2010), we can use the supervised training data we have estimated as input to a supervised learning algorithm for training a language generator.

7 Conclusions

We have presented a novel system that learns a semantic parser for interpreting navigation instructions by simply observing the actions of human followers without using any prior linguistic knowledge or direct supervision. We demonstrated the need to model landmarks when executing longer, more complex instructions. We also introduced a plan refinement algorithm that fairly accurately infers the correct navigation plan specified in the instructions by using a learned semantic lexicon to remove extraneous information. Overall, our approach demonstrates an interesting and novel form of grounded language learning for a complex and useful task.

Acknowledgments

We thank Matt MacMahon for assistance with the original data and the MARCO framework. This work was funded by the NSF grants IIS-0712097 and IIS-1016312. Most of the experiments were run on the Mastodon Cluster, provided by NSF Grant EIA-0303609.

References

Anderson, A.; Bader, M.; Bard, E.; Boyle, E.; Doherty, G. M.; Garrod, S.; Isard, S.; Kowtko, J.; McAllister, J.; Miller, J.; Sotillo, C.; Thompson, H. S.; and Weinert, R. 1991. The HCRC map task corpus. *Language and Speech* 34:351–366.

Bordes, A.; Usunier, N.; and Weston, J. 2010. Label ranking under ambiguous supervision for learning semantic correspondences. In *ICML-2010*.

Branavan, S.; Chen, H.; Zettlemoyer, L. S.; and Barzilay, R. 2009. Reinforcement learning for mapping instructions to actions. In *ACL-09*.

Chen, D. L.; Kim, J.; and Mooney, R. J. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research* 37:397–435.

Kate, R. J., and Mooney, R. J. 2006. Using string-kernels for learning semantic parsers. In *ACL-06*, 913–920.

Kate, R. J., and Mooney, R. J. 2007. Learning language semantics from ambiguous supervision. In *AAAI-2007*, 895–900.

Kim, J., and Mooney, R. J. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *COLING-10*.

Klein, W. 1982. Local deixis in route directions. In Jarvella, R. J., and Klein, W., eds., *Speech, Place, and Action: Studies in Deixis and Related Topics*. Wiley. 161–182.

Kollar, T.; Tellex, S.; Roy, D.; and Roy, N. 2010. Toward understanding natural language directions. In *HRI-2010*.

Lau, T.; Drews, C.; and Nichols, J. 2009. Interpreting written how-to instructions. In *IJCAI-09*.

Liang, P.; Jordan, M. I.; and Klein, D. 2009. Learning semantic correspondences with less supervision. In *ACL-09*.

Lu, W.; Ng, H. T.; Lee, W. S.; and Zettlemoyer, L. S. 2008. A generative model for parsing natural language to meaning representations. In *EMNLP-08*.

MacMahon, M.; Stankiewicz, B.; and Kuipers, B. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *AAAI-2006*.

Matuszek, C.; Fox, D.; and Koscher, K. 2010. Following directions using statistical machine translation. In *HRI-2010*.

Mooney, R. J. 2008. Learning to connect language and perception. In *AAAI-08*, 1598–1601.

Shimizu, N., and Haas, A. 2009. Learning to follow navigational route instructions. In *IJCAI-09*.

Siskind, J. M. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition* 61(1):39–91.

Snyder, B., and Barzilay, R. 2007. Database-text alignment via structured multilabel classification. In *IJCAI-07*.

Thompson, C. A., and Mooney, R. J. 2003. Acquiring word-meaning mappings for natural language interfaces. *JAIR* 18:1–44.

Vogel, A., and Jurafsky, D. 2010. Learning to follow navigational directions. In *ACL-10*.

Wong, Y. W., and Mooney, R. 2006. Learning for semantic parsing with statistical machine translation. In *HLT-NAACL-06*, 439–446.

Wunderlich, D., and Reinelt, R. 1982. How to get there from here. In Jarvella, R. J., and Klein, W., eds., *Speech, Place, and Action: Studies in Deixis and Related Topics*. Wiley. 183–202.

Zettlemoyer, L. S., and Collins, M. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.