The Dissertation Committee for David Lieh-Chiang Chen
certifies that this is the approved version of the following dissertation:

# Learning Language from Ambiguous Perceptual Context

Committee:

Raymond J. Mooney, Supervisor

Regina Barzilay

Katrin Erk

Kristen Grauman

Peter Stone

# Learning Language from Ambiguous Perceptual Context

by

**David Lieh-Chiang Chen, B.S.; M.S.C.S.**

**DISSERTATION**

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

**DOCTOR OF PHILOSOPHY**

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2012

Dedicated to my loving family.

# Acknowledgments

They say getting a PhD is a lot like running a marathon. It is a long process with many ups and downs. Fortunately I have had a lot of help along the way, both when I ran the Austin marathon and in obtaining my degree.

The person that has been the most influential in my research career is my advisor Ray Mooney. Over my six years at UT, he has guided me and supported me through the good times and the bad times. I am thankful I found an advisor who not only shares my research interests, but whom I have fun working with. I enjoyed my weekly meetings with Ray where we discussed topics ranging from research ideas to movie quotes. Most of the ideas in this dissertation have been hatched during those meetings. I appreciate Ray's patience with me during those years when nothing seemed to be working. It has been a long journey, but I am glad I was able to become Ray's 24th academic offspring.

I would like to thank my committee members—Regina Barzilay, Kristen Grauman, Katrin Erk, and Peter Stone—for their feedback and suggestions that helped improve this dissertation considerably. I am thankful for all my discussions with Regina that pushed me to look at the broader picture and compare and contrast the different approaches to the language grounding problem. I am also thankful for Kristen who has been instrumental in my job search process.

Another person that has helped me greatly is my mentor Bill Dolan from when I interned at Microsoft Research. My internship project of collecting translation and paraphrase data using videos was both novel and fun to work on. Even after the internship Bill has continually helped me both when we were putting together a conference paper and when I was looking for a job.

I want to thank past and present members of the Machine Learning group whom I have worked with. I want to thank John Wong, Rohit Kate, Razvan Bunescu, and Lilyana Mihalkova for their frequent help and guidance early in my PhD career. I also want to thank Joohyun Kim, Sonal Gupta, and Lu Guo for our collaborations and discussions about research ideas. I enjoyed my conversations with my officemates Tuyen Huynh and Sindhu Raghavan which provided welcomed breaks between long hours of work. Finally, I want to thank the rest of the group: Yinon Bentor, Bishal Barman, Dan Garrette, Joe Reisinger, Ayan Acharya, Hyeonseo Ku, Tanvi Motwani, Parag Singla and Karl Pichotta for attending my practice talks and giving me feedback about my research and presentations.

During my time in Austin, I have also made many friends including Edmund Wong, Eric Rozner, Ricky Chang, Ian Wehrman, Sangmin Lee, Nalini Belaramani, Sungju Hwang and Ben Delaware in the department. They have made my six years here a fun experience and helped me through tough times at school.

I would also like to thank members of the UTCS department staff who have helped me with various issues over the last six years. In particular, Katherine Utz, Stacy Miller, Gloria Ramirez, and Lydia Griffith have been incredibly resourceful and very prompt in helping me with numerous administrative issues.

I want to thank the UTCS department and the National Science Foundation (NSF) for funding me for six years. The research in this thesis was supported by the NSF grants IIS-0712097 and IIS-1016312.

Finally, I want to thank my family including my parents, my brother Leo and my wife Emma. I would not have been able to complete my PhD without their loving support. I want to especially thank Emma for being an understanding wife and for sharing the ups and downs of my PhD journey.

David Lieh-Chiang Chen

*The University of Texas at Austin*
*May 2012*

# Learning Language from Ambiguous Perceptual Context

Publication No. ______________

David Lieh-Chiang Chen, Ph.D.
The University of Texas at Austin, 2012

Supervisor: Raymond J. Mooney

Building a computer system that can understand human languages has been one of the long-standing goals of artificial intelligence. Currently, most state-of-the-art natural language processing (NLP) systems use statistical machine learning methods to extract linguistic knowledge from large, annotated corpora. However, constructing such corpora can be expensive and time-consuming due to the expertise it requires to annotate such data. In this thesis, we explore alternative ways of learning which do not rely on direct human supervision. In particular, we draw our inspirations from the fact that humans are able to learn language through exposure to linguistic inputs in the context of a rich, relevant, perceptual environment.

We first present a system that learned to sportscast for RoboCup simulation games by observing how humans commentate a game. Using the simple assumption that people generally talk about events that have just occurred, we pair each textual comment with a set of events that it could be referring to. By applying an EM-like algorithm, the system simultaneously learns a grounded language model and aligns each description to the corresponding event. The system does not use any prior language knowledge and was able to learn to sportscast in both English and Korean. Human evaluations of the generated commentaries indicate they are of reasonable quality and in some cases even on par with those produced by humans.

For the sportscasting task, while each comment could be aligned to one of several events, the level of ambiguity was low enough that we could enumerate all the possible alignments. However, it is not always possible to restrict the set of possible alignments to such limited numbers. Thus, we present another system that allows each sentence to be aligned to one of exponentially many connected subgraphs without explicitly enumerating them. The system first learns a lexicon and uses it to prune the nodes in the graph that are unrelated to the words in the sentence. By only observing how humans follow navigation instructions, the system was able to infer the corresponding hidden navigation plans and parse previously unseen instructions in new environments for both English and Chinese data. With the rise in popularity of crowdsourcing, we also present results on collecting additional training data using Amazon's Mechanical Turk. Since our system only needs supervision in the form of language being used in relevant contexts, it is easy for virtually anyone to contribute to the training data.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Being able to communicate with a computer in human languages is one of the ultimate goals of artificial intelligence (AI) research. Instead of learning special commands or control sequences (e.g. a series of mouse clicks, typing, or gestures), we could articulate what we want in our own words. In response, the computer could also present information to us or ask questions verbally without those responses having been programmed into the system. In order to achieve this goal, there are two tasks the computer must become competent at: the ability to interpret human languages and the ability to generate coherent natural language content. The research areas of *semantic parsing* (Mooney, 2007; Zettlemoyer & Collins, 2007; Lu, Ng, Lee, & Zettlemoyer, 2008; Kwiatkowski, Zettlemoyer, Goldwater, & Steedman, 2010; Liang, Jordan, & Klein, 2011; Goldwasser, Reichart, Clarke, & Roth, 2011) and *natural language generation (NLG)* (Barzilay & Lee, 2002; Duboue & McKeown, 2003; Barzilay & Lapata, 2005; Wong & Mooney, 2007; Lu, Ng, & Lee, 2009; Angeli, Liang, & Klein, 2010) aim to solve these two tasks, respectively.

Formally, semantic parsing is the task of translating natural language sentences into formal representations of their meanings. The choice of the representation language depends on the specific application domain and can range from predicate logic to SQL statements to any other formal language that supports automated reasoning. There are typically two parts to building a semantic parser. One is building a lexicon that defines the meanings of words or short phrases. The other part is building compositional rules that successively combine smaller meaning representations into larger, coherent representations of complete sentences.

Natural language generation is the reciprocal task where the goal is to generate natural language content from formal representations. However, unlike semantic parsing where the correct output is usually well-defined, generally there are many acceptable outputs for a NLG system due to the expressiveness of natural languages. In addition to being able to translate a formal representation into natural language sentences (*surface realization*) (Barzilay & Lee, 2002; Wong & Mooney, 2007; Lu et al., 2009), a NLG system must also decide what information to express (*content selection*) (Duboue & McKeown, 2003; Zaragoza & Li, 2005; Barzilay & Lapata, 2005).[1]

The traditional approach to building systems that learn to perform semantic parsing and NLG requires careful semantic annotations of natural language sentences (Ge & Mooney, 2005; Kate & Mooney, 2007; Wong & Mooney, 2007; Zettlemoyer & Collins, 2007; Lu et al., 2008). A typical training

[1]Here we are only looking at generating a single sentence at a time and do not consider other NLG issues such as making the content of a longer document coherent or merging multiple sentences to make it more readable.

corpus would consist of hundreds or thousands of natural language sentences paired with their semantic representations. While supervised training in this manner has been shown to work well, the effort required to construct such training corpora is non-trivial. Performing these semantic annotations is akin to translation where the annotator must be fluent in both the natural language and the formal language being used.

In order to reduce the amount of supervision required to train these systems, we draw our inspirations from how children learn language. Clearly children do not learn language from carefully annotated corpora. Instead, they learn to connect words and phrases to objects and events in the world through simultaneous exposure to linguistic inputs and their perceptual environment. While we are not trying to emulate exactly how a child learns language, we note that this form of training data is more natural and does not require human annotations of semantics. Thus, our goal is to build a system that would observe language being used in relevant perceptual contexts, and learn the correspondences between language and objects/actions/events in the world. In this manner, the semantics of languages is grounded in the perceptual experience of the world (Harnad, 1990).

This type of *grounded language acquisition* sits at the intersection of many different subareas of AI, including natural language processing, robotics, cognitive science, and computer vision. Consequently, there has been many different approaches to this problem, coming from different research perspectives (Bailey, Feldman, Narayanan, & Lakoff, 1997; Roy, 2002; Barnard, Duygulu, Forsyth, de Freitas, Blei, & Jordan, 2003; Yu & Ballard, 2004; Gold & Scassellati, 2007; Fleischman & Roy, 2008; Branavan, Chen, Zettlemoyer, & Barzilay, 2009; Liang et al., 2009; Vogel & Jurafsky, 2010; Feng & Lapata, 2010; Branavan, Silver, & Barzilay, 2011; Tellex, Kollar, Dickerson, Walter, Banerjee, Teller, & Roy, 2011). Compared to most of the earlier work which focus on building systems to accomplish a particular real-world task, we are more concerned with learning the semantics of language, using the specific applications only to demonstrate the results of the language learning. Our goal is to learn not only just meanings of words and phrases, but also how to compose them together to form complete meanings. In this regard, we also take a language-agnostic approach by disallowing any language-specific resources in our systems. While this usually makes learning more difficult, it also ensures that our system can work across different languages. Given that our focus is on language learning, we do not attempt to solve the computer vision or robotics problems, opting to study the problem in simulated environments instead. Nevertheless, we tried to retain many important properties of a dynamic world with multiple agents and actions while avoiding the complexities of real-world perceptions.

The central question this thesis tries to address is how to solve the reference ambiguity problem. In order to utilize the perceptual environment as a form of semantic supervision, we need to determine which parts of the world the speaker is referring to. This relates to the indeterminacy of translation, or "gavagai" problem raised by Quine (1960). In his book, Quine described the problem a field linguist faces trying to learn an unknown language from the natives. If the native uttered “gavagai” upon seeing a rabbit run past, it would be natural to think that “gavagai” means rabbit. However, other translations would also be consistent with the evidence: “Something is running”, “Let’s go hunting”, “An animal”, “An undetached rabbit-part”, etc. Similarly, our system faces the same problem trying to establish correspondences between language and the observed perceptual contexts. From all the perceptual data that co-occurred with the language, the system must decide which objects, actions, or events were being referred to. In some cases, the utterance might not even be connected to any of the perceptions. The system would thus also have to decide if an alignment should be made at all.

The reference ambiguity problem is difficult to solve and humans rely on other cues such as gaze following and the theory of mind to help resolve the ambiguity. Thus, instead of presenting the computer system with raw sensory data, we define for the computer system the space of possible alignments to make the problem more manageable. The size of this space depends on the complexity of the domain and the level of granularity of the formal representations of meanings.

We first present a system that deals with small amounts of ambiguity: each natural language sentence can only be aligned to a handful of events. We study this system in the context of learning how to sportscast a RoboCup simulation game. The system observes how humans commentate the games and try to align each textual comment to one of the events that has just occurred. While performing the alignment, the system also learns a translation model between the natural language comments and the representations of the events. Using an EM-like approach, the system alternates between aligning the data and building the translation model. The learned translation model allows the system to describe any events defined by the formal representations. The system is limited in its ability to deal with reference ambiguity because it enumerates all the possible alignments and computes a score for each one. Thus, the time complexity grows at least linearly with the number of possible alignments (there is also additional cost in training the translation model). We experimented with different initial alignments as well as different scoring functions for selecting the best alignments according to our translation model. Overall, the best system with no initial alignments was able to correctly align about 70% of the sentences for both the English and Korean data we collected. To produce an actual sportscast, we also developed a content selection algorithm for learning which events to describe at any given time. Evaluation of the overall sportscasting system was performed using crowdsourcing. In most cases, our system produced reasonable sportscasts and sometimes even on par with those produced by humans.

The second system we present can deal with a much higher degree of reference ambiguity. Instead of a list of potential meanings for each sentence, we represent the space of possible meanings as a connected graph. Consequently, each of the sentences can now be aligned to one of exponentially many connected subgraphs. The graphical representation is often more natural because

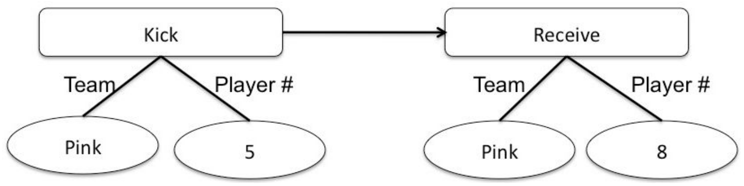


Figure 1.1: Graphical representation of a passing event from Pink player #5 to Pink player #8

entities in the ambiguous context are usually related. For example, in the sportscasting domain we had two separate events that represented kicking the ball and passing the ball. However, passing the ball includes kicking the ball as part of the action. Using a graph we could instead represent passing as a kicking event followed by a receiving event as shown in Figure 1.1. We study this more advanced system in the context of learning to interpret natural language navigation instructions in a virtual environment. The system observes how humans follow navigation instructions and infers the correct navigation plans specified by those instructions. The space of possible plans is a connected graph with the edges representing temporal orders and parameters of actions. After the system has determined the navigation plans associated with each instruction in the training data, it then learns a semantic parsing model which can be used to interpret novel instructions. During testing, the parsed plans are carried out by a virtual robot for the end-to-end task evaluation. The system was able to infer close to 80% of the human-annotated navigation plans for both the English and Chinese data. Moreover, it reached the correct destination over half the time for interpreting a single sentence in an instruction. However, the task completion rate is much lower when evaluated on longer instructions consisting of several sentences and the overall performance is still worse than a previous, hand-coded system (MacMahon et al., 2006).

Given that one of the motivations for doing grounded language learning is to lessen the efforts required to collect the training data, we demonstrate the ease of labeling new data for the navigation domain through an interactive application. The human annotator can either provide new navigation instructions for a specified path, or follow an existing instruction in the virtual environment. The first task generates potential new instructions for the system to train or evaluate on, and the second task verifies the qualities of those instructions. Neither of these tasks require knowledge about the formal representations and can be done by a layperson. Using Amazon's Mechanical Turk, we collected over 2,400 instructions and 14,000 follower traces in two months for under $300. After filtering for quality using a strict majority agreement, there were over 1,000 good instructions. This pilot data collection suggests that a non-expert is indeed capable of providing useful training data.

## 1.1 Thesis Contributions

This thesis makes two main contributions to the area of grounded language acquisition. First, we defined two tasks that serve as testbeds for systems that aim to learn from ambiguous supervision. We made available to the public all the data we collected and annotated. Given that this a relatively young area of research with no standard datasets for training and testing, we aimed to contribute to the pool of common resources people could use to compare relative system performance. The sportscasting data has already been adopted by other researchers for evaluating their algorithms (Liang et al., 2009; Bordes, Usunier, & Weston, 2010; Angeli et al., 2010; Hajishirzi, Hockenmaier, Muellar, & Amir, 2011). Moreover, the tasks we defined are end-to-end tasks that could be broken up into many subtasks. For example, the navigation task could be broken up into three smaller tasks: inferring navigation plans for the instructions in the training data, building a semantic parser for interpreting navigation instructions, and building a robot (physical or virtual) for carrying out navigation plans. This allows researchers interested in different subproblems to concentrate on a particular portion of the problem while working within the larger framework.

The second contribution we make is presenting two algorithms for solving the reference ambiguity problem. The first algorithm learns to align a natural language sentence to a formal representation of meaning out of a small list of possible meanings. We experimented with different ways of utilizing various semantic parsing and natural language generation models for selecting the most likely representation. The second algorithm bypasses the need to enumerate the set of meanings and consequently can deal with much larger spaces of possible alignments. By first learning a lexicon that maps words and short phrases to their meanings, the system can determine the most likely meaning of a sentence by inspecting a part of the sentence at a time. We demonstrated how the system can map each sentence to a connected subgraph from a graphical representation of the space of possible meanings. Finally, we also built complete end-to-end systems that combine our algorithms with off-the-shelf components to solve the two complete tasks (sportscasting and navigation) that we have defined.

## 1.2 Thesis Outline

The remainder of the thesis is organized as follows.

- Chapter 2 reviews previous work this thesis directly builds upon as well as terminologies and notations that are used throughout this thesis.
- Chapter 3 presents our framework for learning from limited ambiguity and its application to the sportscasting task.
- Chapter 4 describes our more advanced framework that deals with ambiguous relational data. We also present the result of using this framework to learn to interpret navigation instructions and how to collect additional training data for this task using Mechanical Turk.
- Chapter 5 reviews related work in grounded language acquisition, ambiguously supervised learning, semantic parsing, and NLG.
- Chapter 6 discusses future work and chapter 7 concludes the thesis.

We note that some the material presented in Chapter 3 and Chapter 4 have appeared in our previous publications (Chen & Mooney, 2008; Chen, Kim, & Mooney, 2010; Chen & Mooney, 2011).

# Chapter 2

# Background

In this chapter we will discuss previous work that this thesis directly builds upon. In particular, we will describe a couple of supervised learning systems for semantic parsing and natural language generation. These systems assume training data in the form of parallel corpora where each natural language sentence is aligned with its correct formal representation. Our work extends these existing techniques to deal with ambiguous training data where the alignments to the correct representations are unknown and only the space of potential alignments are given. While we concentrate on describing only a handful of related pieces of work in this chapter to give the readers sufficient background to understand the rest of the thesis, we will have a more detailed review of relevant research in robotics, computer vision, computational linguistics and machine learning in Chapter 5.

The goal of this thesis can be formally described as learning how to build semantic parsers and language generators from perceptual data. Semantic parsers map natural language (NL) sentences to *meaning representations* (MRs) in some formal logical language. On the other hand, language generators performs the reverse mapping (MRs to NL). Existing work has mostly focused on learning from supervised corpora in which each sentence is manually annotated with its correct MR (Mooney, 2007; Zettlemoyer & Collins, 2007; Lu et al., 2008; Kwiatkowski et al., 2010). Such human annotated corpora are expensive and difficult to produce, limiting the utility of this approach.

Kate and Mooney (2007) introduced an extension to one such such system, KRISP (Kate & Mooney, 2006), so that it can learn from ambiguous training data that requires little or no human annotation effort. However, the resulting system , KRISPER, is unable to *generate* language or scale to high levels of ambiguity.

Since our sportscasting task requires language generation, we enhanced another system called WASP (Wong & Mooney, 2006) that is capable of language generation as well as semantic parsing. We briefly describe these existing systems below. All of them assume they have access to a formal deterministic *context-free grammar* (CFG) that defines the formal *meaning representation language* (MRL). Since MRLs are formal computer-interpretable languages, such grammars are usually readily available.

## 2.1 KRISP and KRISPER

KRISP (Kernel-based Robust Interpretation for Semantic Parsing) (Kate & Mooney, 2006) uses *support vector machines* (SVMs) with string kernels to build semantic parsers. SVMs are state-of-the-art machine learning methods that learn maximum-margin separators to prevent over-fitting in very high-dimensional data such as natural language text (Joachims, 1998). They can be extended to non-linear separators and non-vector data by exploiting *kernels* that implicitly create an even higher dimensional space in which complex data is (nearly) linearly separable (Shawe-Taylor & Cristianini, 2004). Kernels over strings and trees have been effectively applied to a variety of problems in text learning and NLP (Lodhi, Saunders, Shawe-Taylor, Cristianini, & Watkins, 2002; Zelenko, Aone, & Richardella, 2003; Collins, 2002; Bunescu & Mooney, 2005). In particular, KRISP uses the string kernel introduced by Lodhi et al. (2002) to classify substrings in a NL sentence.

First, KRISP learns classifiers that recognize when a word or phrase in an NL sentence indicates that a particular concept in the MRL should be introduced into its MR. It uses production rules in the MRL grammar to represent semantic concepts, and it learns classifiers for each production that classify NL substrings as indicative of that production or not. When semantically parsing a sentence, each classifier estimates the probability of each production covering different substrings of the sentence. This information is then used to compositionally build a complete MR for the sentence. Given the partial matching provided by string kernels and the over-fitting prevention provided by SVMs, KRISP has been experimentally shown to be particularly robust to noisy training data (Kate & Mooney, 2006).

KRISPER (Kate & Mooney, 2007) is an extension to KRISP that handles ambiguous training data in which each sentence is annotated with a *set* of potential MRs, only one of which is correct. Psuedocode for KRISPER is shown in Algorithm 1. It employs an iterative approach analogous to *expec-*

**Algorithm 1** Krisper

```
input sentences $S$ and their associated sets of meaning representations $MR(s)$
output $BestExamplesSet$, a set of NL-MR pairs,
    $SemanticModel$, a Krisp semantic parser
 1:
 2: main
 3:   //Initial training loop
 4:   for sentence $s_i \in S$ do
 5:     for meaning representation $m_j \in MR(s_i)$ do
 6:       add $(s_i, m_j)$ to $InitialTrainingSet$
 7:     end for
 8:   end for
 9:   $SemanticModel$ = Train($InitialTrainingSet$)
10:
11:   //Iterative retraining
12:   repeat
13:     for sentence $s_i \in S$ do
14:       for meaning representation $m_j \in MR(s_i)$ do
15:         $m_j.score = Evaluate(s_i, m_j, SemanticModel)$
16:       end for
17:     end for
18:     $BestExampleSet \leftarrow$ The set of consistent examples $T = \{(s,m)|s \in$
        $S, m \in MR(s)\}$ such that $\sum_T m.score$ is maximized
19:     $SemanticModel = Train(BestExamplesSet)$
20:   until Convergence or MAX_ITER reached
21: end main
22:
23: function Train($TrainingExamples$)
24:   Train Krisp on the unambiguous $TrainingExamples$
25:   return The trained Krisp semantic parser
26: end function
27:
28: function Evaluate($s$, $m$, $SemanticModel$)
29:   Use the Krisp semantic parser $SemanticModel$ to find a derivation of
      meaning representation $m$ from sentence $s$
30:   return The parsing score
31: end function
```

*tation maximization* (EM) (Dempster, Laird, & Rubin, 1977) that improves upon the selection of the correct NL–MR pairs in each iteration. In the first iteration (lines 3-9), it assumes that all of the MRs paired with a sentence are correct and trains KRISP with the resulting noisy supervision. In subsequent iterations (lines 11-27), KRISPER uses the currently trained parser to score each potential NL–MR pair, selects the most likely MR for each sentence, and retrains the parser on the resulting disambiguated supervised data. In this manner, KRISPER is able to learn from the type of weak supervision expected for a grounded language learner exposed only to sentences in ambiguous contexts. However, the system has previously only been tested on supervised data with randomly added ambiguities or artificially generated data.

## 2.2 WASP and WASP$^{-1}$

WASP (Word-Alignment-based Semantic Parsing) (Wong & Mooney, 2006) uses state-of-the-art *statistical machine translation* (SMT) techniques (Brown, Cocke, Della Pietra, Della Pietra, Jelinek, Lafferty, Mercer, & Roossin, 1990; Yamada & Knight, 2001; Chiang, 2005) to learn semantic parsers. SMT methods learn effective machine translators by training on *parallel corpora* consisting of human translations of documents into one or more alternative natural languages. The resulting translators are typically significantly more effective than manually developed systems and SMT has become the dominant approach to machine translation. WASP adapted such methods to learn to translate from NL to MRL rather than from one NL to another.

First, an SMT *word alignment* system, GIZA++ (Och & Ney, 2003; Brown, Della Pietra, Della Pietra, & Mercer, 1993), is used to acquire a bilingual lexicon consisting of NL substrings coupled with their translations in the target MRL. As formal languages, MRLs frequently contain many purely syntactic tokens such as parentheses or brackets which are difficult to align with words in NL. Consequently, WASP aligns words in the NL with productions of the MRL grammar used in the parse of the corresponding MR. Therefore, GIZA++ is used to produce an N to 1 alignment between the words in the NL sentence and a sequence of MRL productions corresponding to a top-down left-most derivation of the corresponding MR.

Complete MRs are then formed by combining these NL substrings and their translations using a grammatical framework called *synchronous CFG* (SCFG) (Aho & Ullman, 1972), which forms the basis of most existing *syntax-based* SMT (Yamada & Knight, 2001; Chiang, 2005). In a SCFG, the right hand side of each production rule contains *two* strings, in this case one in NL and the other in MRL. Derivations of the SCFG simultaneously produce NL sentences and their corresponding MRs. The bilingual lexicon acquired from word alignments over the training data is used to construct a set of SCFG production rules. A probabilistic parser is then produced by training a maximum-entropy model using EM to learn parameters for each of these SCFG productions (Riezler, Prescher, Kuhn, & Johnson, 2000; Zettlemoyer & Collins, 2005). To translate a novel NL sentence into its MR, a probabilistic chart parser (Stolcke, 1995) is used to find the most probable synchronous derivation that generates the given NL, and the corresponding MR generated by this derivation is returned.

Since SCFGs are symmetric, they can be used to *generate* NL from MR as well as parse NL into MR (Wong & Mooney, 2007). This allows the same learned grammar to be used for both parsing and generation, an elegant property that has important advantages (Shieber, 1988). The generation system, WASP$^{-1}$, uses a *noisy-channel model* (Brown et al., 1990):

$$\operatorname{argmax}_{\mathbf{e}} \Pr(\mathbf{e}|\mathbf{f}) = \operatorname{argmax}_{\mathbf{e}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e}) \tag{2.1}$$

Where $\mathbf{e}$ refers to the NL string generated for a given input MR, $\mathbf{f}$. $\Pr(\mathbf{e})$ is the *language model*, and $\Pr(\mathbf{f}|\mathbf{e})$ is the *parsing model* provided by WASP's learned SCFG. The generation task is to find a sentence $\mathbf{e}$ such that (1) $\mathbf{e}$ is a good sentence a priori, and (2) its meaning is the same as the input MR. For the language model, WASP$^{-1}$ uses a standard $n$-gram model, which is useful in ranking candidate generated sentences (Knight & Hatzivassiloglou, 1995).

Since both WASP and WASP$^{-1}$ learn the same underlying SCFG rules, we will refer to both system jointly as WASP in the rest of this thesis.

# Chapter 3

# Learning from Supervision with Limited Ambiguity

In this chapter we will look at the language learning problem when there is limited ambiguity in the training data. In particular, we will examine this problem in the context of building a sportscasting application. The goal of the application is to generate NL descriptions of events as they occur in a simulated soccer game. The training data consists of a stream of descriptive textual comments generated by humans and automatically extracted events from the simulation. We present a framework that simultaneously establishes correspondences between the individual comments and the events they are describing as well as builds a translation model that supports both semantic parsing and natural language generation. We also present a novel algorithm for selecting the relevant events to describe during the course of a game.

We evaluated each component of our system individually as well as the overall system. Human judges recruited over the Internet indicated that the sportscasts generated by our system are of reasonable quality and in some cases even on par with those produced by humans for our limited domain. We evaluated our system on both English and Korean data with similar results.

This demonstrates that our system is flexible in adapting to learning different natural languages since the system does not rely on any prior linguistic knowledge.

## 3.1 Overview

Our overall goal is to build systems that can learn the semantics of language from its usage in a relevant perceptual context without any direct supervision. This is a difficult task even for humans. Typically a child would be exposed to language specifically geared towards teaching them the meanings of words or phrases (e.g. when a care-giver points to an object and says the name of the object). Since we assume the system has no knowledge of the language to start with, the training data must be closely correlated with the perceptual environment to make this task possible. One source of such data comes from descriptive languages where the speaker gives a literal description of what can or should be seen. Some examples include captions of photos, scripts of movies, audio descriptions (additional narration track for blind and visually impaired consumers of visual media), product descriptions (e.g. clothing), and play-by-play calls of a sports game. For our initial exploration of the ambiguous language learning problem, we chose to use sportscasts as our training data. By observing how humans describe events in a simulated soccer game, our system is able to learn to generate its own sportscast for previously unseen games. A screenshot of our system with generated commentary is shown in Figure 3.1.

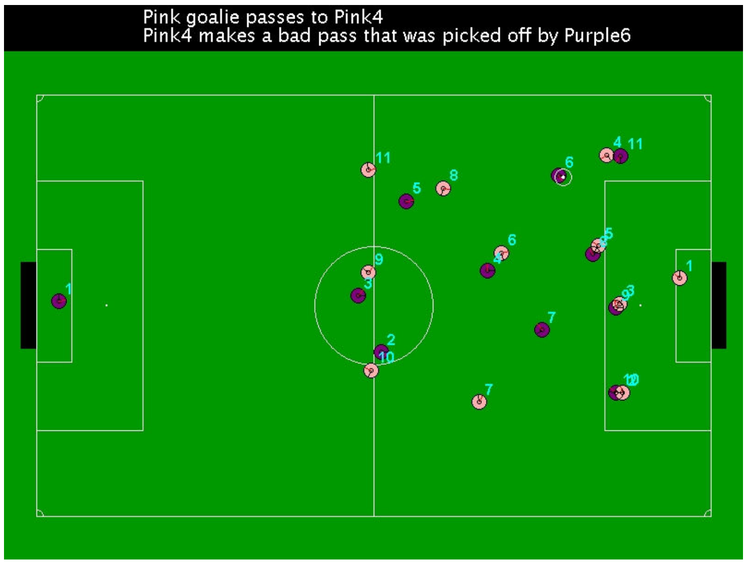

Pink goalie passes to Pink4
Pink4 makes a bad pass that was picked off by Purple6

Figure 3.1: Screenshot of our sportscasting system

It is important to note that real sportscasts typically include a play-by-play commentator and a color commentator who fill in the gaps when no actions are taking place. Since we are concerned with language that is connected to the actions being observed, we concentrate only on the play-by-play portion. We make no attempts at generating a realistic sportscast that contains background information, opinions, and comments that make the sportscast interesting and engaging.

Ideally, our system would be able to process raw visual information and identify relevant objects and events automatically. However, to avoid the complexity of solving the computer vision issues, we studied the problem in a simulated environment instead. Nevertheless, we aimed to retain many of the important properties of a dynamic world with multiple agents and actions. Specifically, we use the RoboCup simulator (Chen, Foroughi, Heintz, Kapetanakis, Kostiadis, Kummeneje, Noda, Obst, Riley, Steffens, Wang, & Yin, 2003) which provides a fairly detailed physical simulation of robot soccer. While several groups have constructed RoboCup commentator systems (André, Binsted, Tanaka-Ishii, Luke, Herzog, & Rist, 2000) that provide textual NL transcripts of simulated games, those systems all use manually-developed templates and do not learn from data.

Our commentator system learns to semantically interpret and generate language in the RoboCup soccer domain by observing an on-going commentary of the game paired with the evolving simulator state. By exploiting existing techniques for abstracting a symbolic description of the activity on the field from the detailed states of the physical simulator (André et al., 2000), we obtain a pairing of natural language with a symbolic description of the perceptual context in which the comment was uttered. However, such training data is ambiguous because each comment usually co-occurs with several events in the game. We integrate and enhance existing methods for learning semantic parsers and NL generators (Kate & Mooney, 2007; Wong & Mooney, 2007) in order to learn to understand and generate language from such ambiguous training data. We also developed a system that, from the same ambiguous training data, learns which events are worth describing, so that it can also perform *content selection*, that is, deciding *what* to say as well as how to say it (*surface realization*).

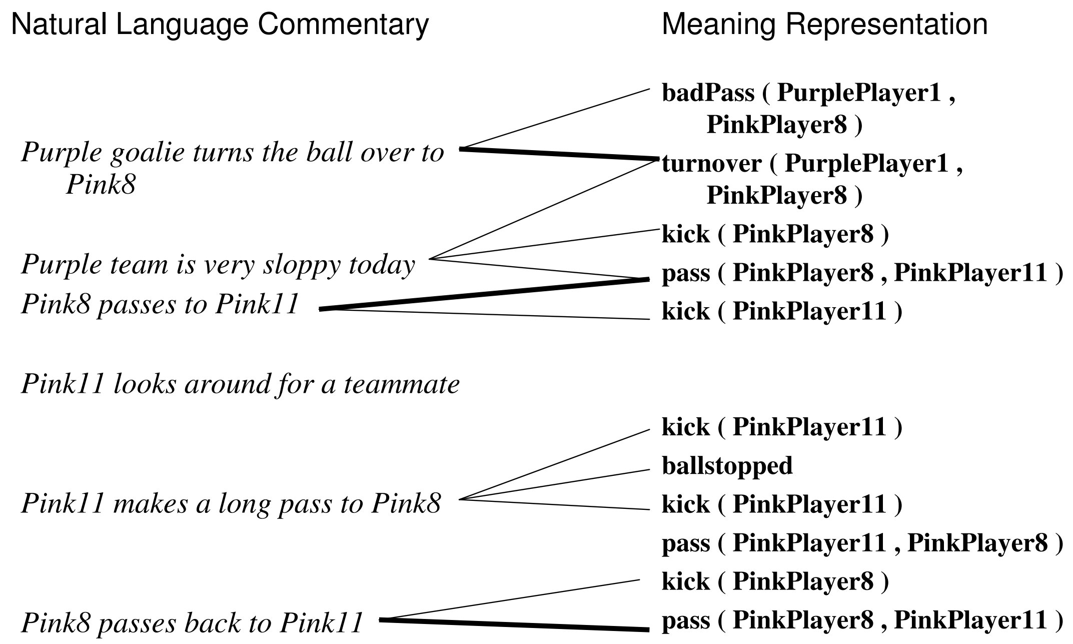
We evaluate our system and demonstrate its language-independence by training it to generate commentaries in both English and Korean. Experiments on test data (annotated for evaluation purposes only) demonstrate that the system learns to accurately semantically parse sentences, generate sentences, and decide which events to describe. Finally, subjective human evaluation of commentated game clips demonstrate that in our limited domain, the system generates sportscasts that are in some cases similar in quality to those produced by humans.

In the rest of this chapter, we will first describe the sportscasting data we collected to train and test our system in Section 3.2. Section 3.3 and Section 3.4 present the details of our framework for learning surface realization and content selection, respectively, as well as experimental evaluations of each component. Section 3.5 presents human evaluations of the overall sportscasting system. We will then discuss a couple of extensions to the basic framework including initializing the system with training data disambiguated by a method proposed by Liang et al. (2009) in Section 3.6 and detecting and removing "superfluous" sentences that do not refer to any extracted events in Section 3.7. Finally, we discuss other potential extensions to the system and summarize the contributions in Section 3.8 and Section 3.9, respectively.

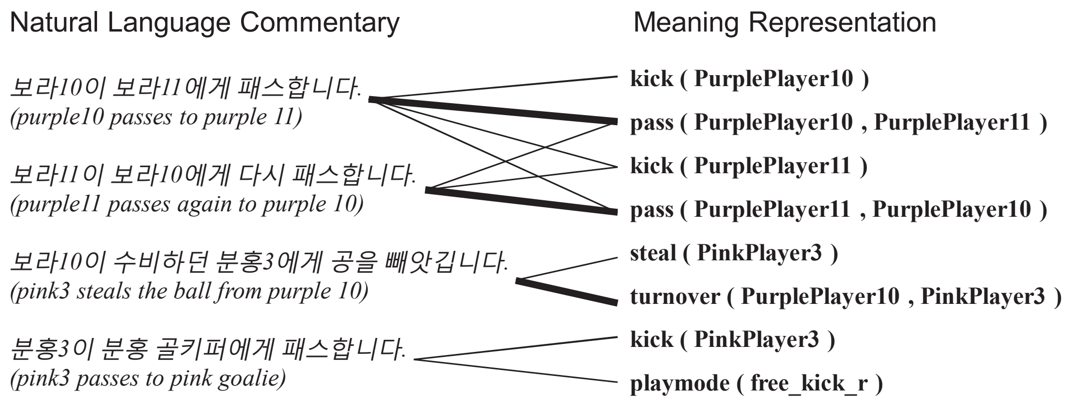
## 3.2 Sportscasting Data

To train and test our system, we assembled human-commentated soccer games from the RoboCup simulation league (www.robocup.org). Since our focus is on language learning and not computer vision, we chose to use simulated games instead of real game videos to simplify the extraction of perceptual information. Based on the Rocco RoboCup commentator's incremental event recognition module (André et al., 2000) we manually developed symbolic representations of game events and a rule-based system to automatically extract them from the simulator traces. The extracted events mainly involve actions with the ball, such as kicking and passing, but also include other game information such as whether the current playmode is kickoff, offside, or corner kick. The events are represented as atomic formulas in predicate logic with timestamps. These logical facts constitute the requisite MRs, and we manually developed a simple CFG for this formal semantic language. Details of the events detected and the complete grammar can be found in Appendix A.

For the NL portion of the data, we had humans commentate games while watching them on the simulator. We collected commentaries in both English and Korean. The English commentaries were produced by two different people while the Korean commentaries were produced by a single person. The commentators typed their comments into a text box, which were recorded with a timestamp. To construct the final ambiguous training data, we paired each comment with all of the events that occurred five seconds or less before the comment was made. Examples of the ambiguous training data are shown in Figure 3.2. The edges connect sentences to events to which they might refer. English translations of the Korean commentaries have been included in the figure for the reader's benefit and are not part of the actual data. Also note that the use of English words for predicates and constants in the MRs


Natural Language Commentary
Meaning Representation
Purple goalie turns the ball over to Pink8
Purple team is very sloppy today
Pink8 passes to Pink11
Pink11 looks around for a teammate
Pink11 makes a long pass to Pink8
Pink8 passes back to Pink11
badPass ( PurplePlayer1 , PinkPlayer8 )
turnover ( PurplePlayer1 , PinkPlayer8 )
kick ( PinkPlayer8 )
pass ( PinkPlayer8 , PinkPlayer11 )
kick ( PinkPlayer11 )
kick ( PinkPlayer11 )
ballstopped
kick ( PinkPlayer11 )
pass ( PinkPlayer11 , PinkPlayer8 )
kick ( PinkPlayer8 )
pass ( PinkPlayer8 , PinkPlayer11 )

(a) Sample trace of ambiguous English training data


Natural Language Commentary
Meaning Representation
보라10이 보라11에게 패스합니다.
(purple10 passes to purple 11)
보라11이 보라10에게 다시 패스합니다.
(purple11 passes again to purple 10)
보라10이 수비하던 분홍3에게 공을 빼앗깁니다.
(pink3 steals the ball from purple 10)
분홍3이 분홍 골키퍼에게 패스합니다.
(pink3 passes to pink goalie)
kick ( PurplePlayer10 )
pass ( PurplePlayer10 , PurplePlayer11 )
kick ( PurplePlayer11 )
pass ( PurplePlayer11 , PurplePlayer10 )
steal ( PinkPlayer3 )
turnover ( PurplePlayer10 , PinkPlayer3 )
kick ( PinkPlayer3 )
playmode ( free_kick_r )

(b) Sample trace of ambiguous Korean training data

Figure 3.2: Examples of our sportscasting training data. Each of the outgoing edges from the comments indicate a potentially associated meaning representation considered by our system. The bold links indicate correct matches between the comments and the meaning representations.

| | English dataset | Korean dataset |
|---|---|---|
| Total # of comments | 2036 | 1999 |
| Total # of words | 11742 | 7941 |
| Vocabulary size | 454 | 344 |
| Avg. words per comment | 5.77 | 3.97 |

Table 3.1: Word statistics for the English and Korean sportscasting datasets

is for human readability only, the system treats these as arbitrary conceptual tokens and must learn their connection to English or Korean words.

We annotated a total of four games which consist of the final matches for the RoboCup simulation league from 2001 to 2004. Word statistics about the data are shown in Table 3.1. While the comments are fairly short due to the nature of sportscasts, this data provides challenges in the form of synonyms (e.g. "Pink1", "PinkG" and "pink goalie" all refer to the same player) and polysemes (e.g. "kick" in "kicks toward the goal" refers to a kick event whereas "kicks to Pink3" refers to a pass event.)

For evaluation purposes only, a gold-standard matching was produced by examining each comment manually and selecting the correct MR if it exists. The matching is only approximate because sometimes the comments contain more information than present in the MRs. For example, a comment might describe the location and the length of a pass while the MR captures only the participants of a pass. The bold lines in Figure 3.2 indicate the annotated correct matches in our sample data. Notice some sentences do not have correct matches. For example, the sentence "Purple team is very sloppy today" in Figure 3.2(a) cannot be represented in our MRL and consequently does not have a corresponding correct MR. For another example, the Korean sentence with the translation “pink3 passes to pink goalie” in Figure 3.2(b) can be represented in our MRL, but does not have a correct match due to the incomplete event detection. A free kick was called while pink3 was passing to the pink goalie so the pass event was not retrieved. Finally, in the case of the sentence “Pink11 makes a long pass to Pink8” in Figure 3.2(a), the correct MR falls outside of the 5-second window.

Alignment statistics for the datasets are shown in Table 3.2. The 2001 final has almost twice the number of events as the other games because it went into double overtime. For each game, the number of NL sentences collected are shown as the total number of comments. Some of these comments are discarded from the training set if they are not associated with any potential MRs (e.g. “Pink11 looks around for a teammate” in Figure 3.2(a)). This happens when no events occur within the 5-second window preceding the comment. The remaining sentences are shown as the number of comments that have MRs. Finally, of these sentences, some of them do not have the correct MR in their set of potential MRs as discussed previously. These essentially become noise in the training data (18% of the English dataset and 8% of the Korean dataset). For example, the 2001 final contains 671 training sentences, but only 520 of them can be aligned correctly. The last part of the table shows the level of ambiguity in the training data. The maximum and the average number of comments associated with each comment are shown as well as the

|  | English dataset |  |  |  | Korean dataset |  |  |  |
|---|---|---|---|---|---|---|---|---|
| Game year | 2001 | 2002 | 2003 | 2004 | 2001 | 2002 | 2003 | 2004 |
| # events | 4003 | 2223 | 2113 | 2318 | 4003 | 2223 | 2113 | 2318 |
| Number of comments |  |  |  |  |  |  |  |  |
| Total | 722 | 514 | 410 | 390 | 673 | 454 | 412 | 460 |
| Have MRs | 671 | 458 | 397 | 342 | 650 | 444 | 396 | 423 |
| Have Correct MR | 520 | 376 | 320 | 323 | 600 | 419 | 369 | 375 |
| Events per comment |  |  |  |  |  |  |  |  |
| Max | 9 | 10 | 12 | 9 | 10 | 12 | 10 | 9 |
| Average | 2.24 | 2.40 | 2.85 | 2.73 | 2.14 | 2.49 | 2.55 | 2.60 |
| Std. Dev. | 1.64 | 1.65 | 2.05 | 1.70 | 2.08 | 3.08 | 3.67 | 2.59 |

Table 3.2: Alignment statistics for the English and Korean datasets. The total numbers of events and comments for each game are shown. The training set only consists of those comments that have at least one MR associated with it. Of the training sentences, 82% of the English dataset and 92% of the Korean dataset contain the correct MRs in their set of potential MRs. The number of events per comment are also shown, with more than two events associated with each comment on average.

standard deviation. On average each comment is associated with more than two possible events so at least 50% of the potential links are incorrect.

## 3.3 Learning Surface Realization from Ambiguous Supervision

While existing systems are capable of solving parts of the sportscasting problem, none of them are able to perform the whole task. We need a system that can both deal with ambiguous supervision like Krisper and generate language like Wasp. We introduce three systems below that can do both. An overview of the differences between the existing systems and the new systems

| Algorithm | Main Learner | Generate | Ambiguous data | Disambiguation criteria |
|---|---|---|---|---|
| KRISP | SVM | No | No | n/a |
| KRISPER | KRISP | No | Yes | KRISP's parse score |
| WASP | GIZA++ & SCFG | Yes | No | n/a |
| WASPER | WASP | Yes | Yes | WASP's parse score |
| KRISPER-WASP | KRISPER & WASP | Yes | Yes | KRISP's parse score |
| WASPER-GEN | WASP | Yes | Yes | NIST score of best NL given MR |

Table 3.3: Overview of the various learning systems presented. The first three algorithms are existing systems. We introduce the last three systems that are able to both learn from ambiguous training data and acquire a language generator. They differ in how they disambiguate the training data.

we present are shown in Table 3.3.

All three systems introduced here are based on extensions to WASP, our underlying language learner. The main problem we need to solve is to disambiguate the training data so that we can train WASP to create a language generator. Each of the new system uses a different disambiguation criteria to determine the best matching between the NL sentences and the MRs.

### 3.3.1 WASPER

The first system is an extension of WASP in a manner similar to how KRISP was extended to create KRISPER. It uses an EM-like retraining to handle ambiguously annotated data, resulting in a system we call WASPER. In general, any system that learns semantic parsers can be extended to handle ambiguous data as long as it can produce confidence levels for given NL–MR pairs. Given a set of sentences $s \in S$ and the set of MRs associated with each sentence $MR(s)$, we disambiguate the data by finding pairs $(s, m)$ such that $s \in S$ and $m = \text{argmax}_{m' \in MR(s)} Pr(m'|s)$. Although probability is used here, a ranking of the relative potential parses would suffice. The pseudocode for WASPER is shown in Algorithm 2. The only difference compared to the KRISPER pseudocode is that we now use a WASP semantic parser instead of a KRISP parser. Also, we produce a WASP language generator as well which is the desired final output for our task.

### 3.3.2 KRISPER-WASP

KRISP has been shown to be quite robust at handling noisy training data (Kate & Mooney, 2006). This is important when training on the very noisy training data used to initialize the parser in KRISPER's first iteration. However, KRISPER cannot learn a language generator, which is necessary for our sportscasting task. As a result, we create a new system called KRISPER-WASP that is both good at disambiguating the training data and capable of generation. We first use KRISPER to train on the ambiguous data and produce a disambiguated training set by using its prediction for the most likely MR for each sentence. This unambiguous training set is then used to train WASP to produce both a parser and a generator.

**Algorithm 2** WASPER

```
input  sentences $S$ and their associated sets of meaning representations $MR(s)$
output  $BestExamplesSet$, a set of NL-MR pairs,
    $SemanticModel$, a WASP semantic parser/language generator
 1: main
 2:   //Initial training loop
 3:   for sentence $s_i \in S$ do
 4:     for meaning representation $m_j \in MR(s_i)$ do
 5:       add $(s_i, m_j)$ to $InitialTrainingSet$
 6:     end for
 7:   end for
 8:   $SemanticModel$ = Train($InitialTrainingSet$)
 9:
10:   //Iterative retraining
11:   repeat
12:     for sentence $s_i \in S$ do
13:       for meaning representation $m_j \in MR(s_i)$ do
14:         $m_j.score = Evaluate(s_i, m_j, SemanticModel)$
15:       end for
16:     end for
17:     $BestExampleSet \leftarrow$ The set of consistent examples $T = \{(s, m)|s \in S, m \in MR(s)\}$ such that $\sum_T m.score$ is maximized
18:     $SemanticModel = Train(BestExamplesSet)$
19:   until Convergence or MAX_ITER reached
20: end main
21:
22: function Train($TrainingExamples$)
23:   Train WASP on the unambiguous $TrainingExamples$
24:   return The trained WASP semantic parser/language generator
25: end function
26:
27: function Evaluate($s$, $m$, $SemanticModel$)
28:   Use the WASP semantic parser in $SemanticModel$ to find a derivation of meaning representation $m$ from sentence $s$
29:   return The parsing score
30: end function
```

### 3.3.3 Wasper-Gen

In both Krisper and Wasper, the criterion for selecting the best NL–MR pairs during retraining is based on maximizing the probability of parsing a sentence into a particular MR. However, since Wasper is capable of both parsing and generation, we could alternatively select the best NL–MR pairs by evaluating how likely it is to *generate* the sentence from a particular MR. Thus, we built another version of Wasper called Wasper-Gen that disambiguates the training data in order to maximize the performance of *generation* rather than parsing. The pseudocode is shown in Algorithm 3. The algorithm is the same as Wasper except for the evaluation function. It uses a generation-based score rather than a parsing-based score to select the best NL–MR pairs.

Specifically, a NL–MR pair $(s, m)$ is scored by computing the NIST score, a machine translation (MT) metric, between the sentence $s$ and the best generated sentence for $m$ (lines 9-12).[1] Formally, given a set of sentences $s \in S$ and the set of MRs associated with each sentence $MR(s)$, we disambiguate the data by finding pairs $(s, m)$ such that $s \in S$ and $m = \text{argmax}_{m' \in MR(s)} NIST(s, \text{argmax}_{s'} Pr(s'|m'))$.

NIST measures the precision of a translation in terms of the proportion of $n$-grams it shares with a human translation (Doddington, 2002). It has also been used to evaluate NL generation. Another popular MT metric is BLEU

[1]A natural way to use a generation-based score would be to use the probability of an NL given a MR ($Pr(s|m)$). However, initial experiments using this metric did not produce very good results. We also tried changing Wasp to maximize the joint probability instead of just the parsing probability. However, this also did not improve the results.

score (Papineni, Roukos, Ward, & Zhu, 2002), but it is inadequate for our purposes because we are comparing one short sentence to another instead of comparing whole documents. BLEU score computes the geometric mean of the $n$-gram precision for each value of $n$, which means the score is 0 if a matching $n$-gram is not found for *every* value of $n$. In the common setting in which the maximum $n$ is 4, any two sentences that do not have a matching 4-gram would receive a BLEU score of 0. Consequently, BLEU score is unable to distinguish the quality of most of our generated sentences since they are fairly short. In contrast, NIST uses an additive score and avoids this problem.

**Algorithm 3** WASPER-GEN

```
input sentences $S$ and their associated sets of meaning representations $MR(s)$
output BestExamplesSet, a set of NL-MR pairs,
    SemanticModel, a WASP semantic parser/language generator
 1: main
 2:     same as Algorithm 2
 3: end main
 4:
 5: function Train(TrainingExamples)
 6:     same as Algorithm 2
 7: end function
 8:
 9: function Evaluate($s$, $m$, SemanticModel)
10:     GeneratedSentence $\leftarrow$ Use the WASP language generator in
        SemanticModel to produce a sentence from the meaning representation
        $m$
11:     return The NIST score between GeneratedSentence and $s$
12: end function
```

### 3.3.4 Experimental Evaluations

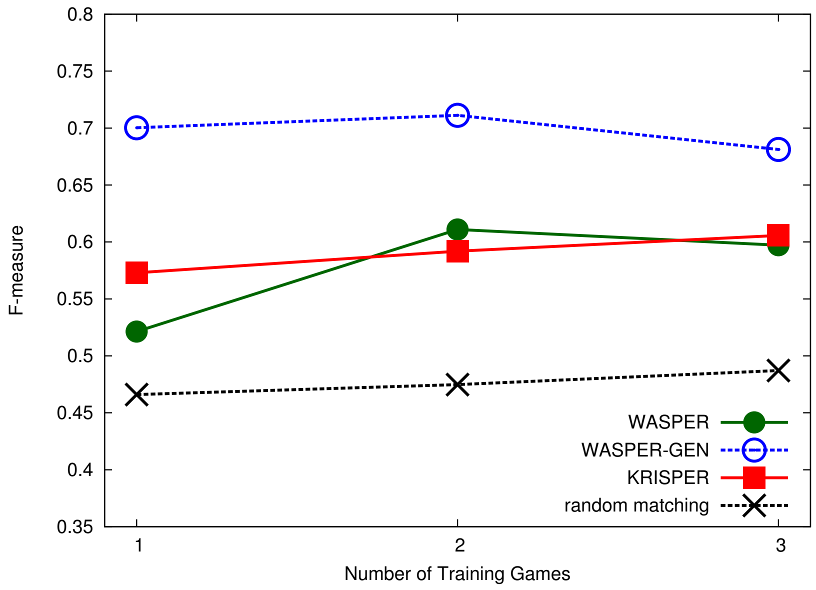
This section presents experimental results on the RoboCup data for four systems: Krisper, Wasper, Krisper-Wasp, and Wasper-Gen. Since we are not aware of any existing systems that could learn how to semantically parse and generate language using ambiguous supervision based on perceptual context, we constructed our own lower and upper baselines using unmodified Wasp. To construct the lower baseline, we randomly pick a meaning for each sentence from its set of potential MRs and trained Wasp on this unambiguous data. We use Wasp trained on the *gold matching* which consists of the correct NL–MR pairs annotated by a human as the upper baseline. This represents an upper-bound on what our systems could achieve if they disambiguated the training data perfectly.

We evaluate each system on three tasks: matching, parsing, and generation. The matching task measures how well the systems can disambiguate the training data. The parsing and generation tasks measure how well the systems can translate from NL to MR, and from MR to NL, respectively.

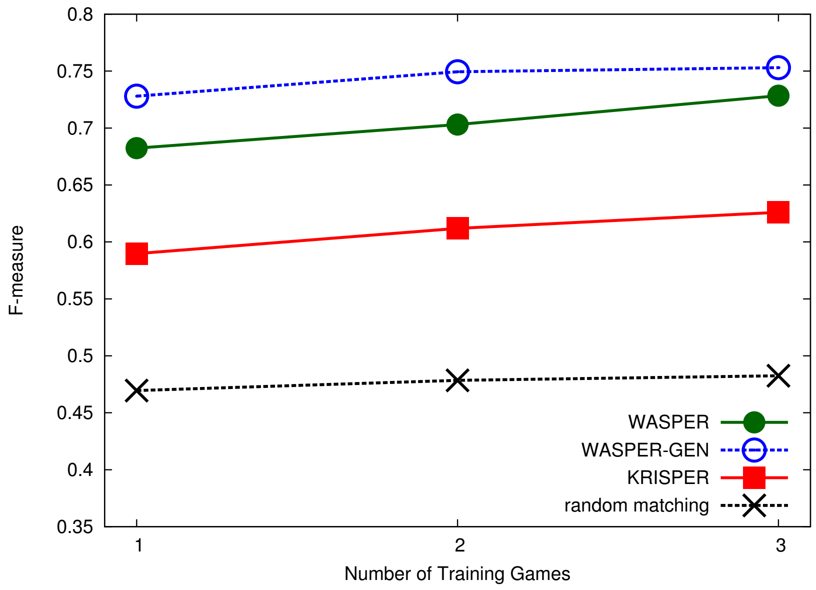
Since there are four games in total, we trained using all possible combinations of one to three games. For matching, we measured the performance on the training data since our goal is to disambiguate this data. For parsing and generation, we tested on the games not used for training. Results were averaged over all train/test combinations. We evaluated matching and parsing using F-measure, the harmonic mean of recall and precision. Precision is the fraction of the system's annotations that are correct. Recall is the fraction of the annotations from the gold-standard that the system correctly produces. Generation is evaluated using BLEU scores which roughly estimates how well the produced sentences match the target sentences. We treat each game as a whole document to avoid the problem of using BLEU score for sentence-level comparisons mentioned earlier. Also, we increase the number of reference sentences for each MR by using all of the sentences in the test data corresponding to equivalent MRs, e.g. if `pass(PinkPlayer7, PinkPlayer8)` occurs multiple times in the test data, all of the sentences matched to this MR in the gold matchings are used as reference sentences for this MR.

#### 3.3.4.1 Matching NL and MR

Since handling ambiguous training data is an important aspect of grounded language learning, we first evaluate how well the various systems pick the correct NL–MR pairs. Figure 3.3 shows the F-measure for identifying the correct set of pairs for the various systems.[2] All of the learning systems perform significantly better than random which has a F-measure below 0.5. For both the English and Korean data, WASPER-GEN is the best system. WASPER also equals or outperforms the previous system KRISPER as well.

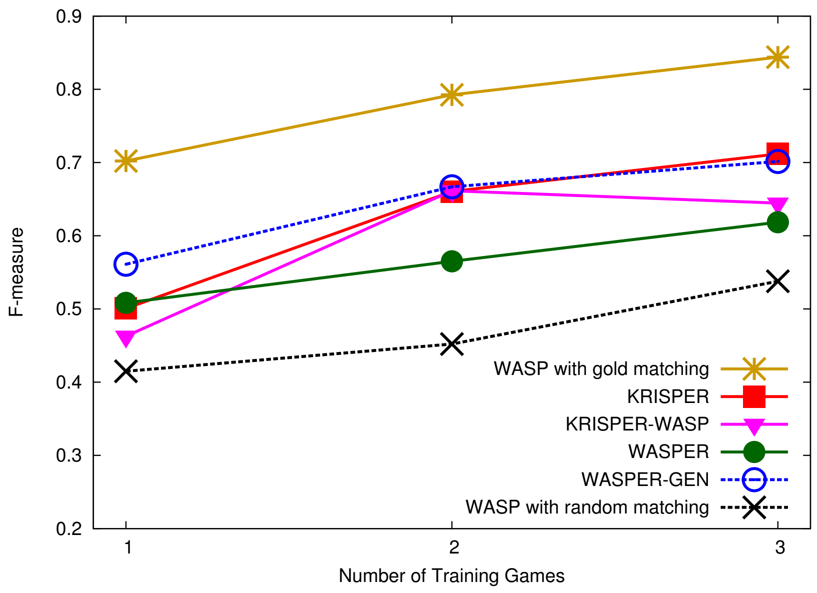


(a) English

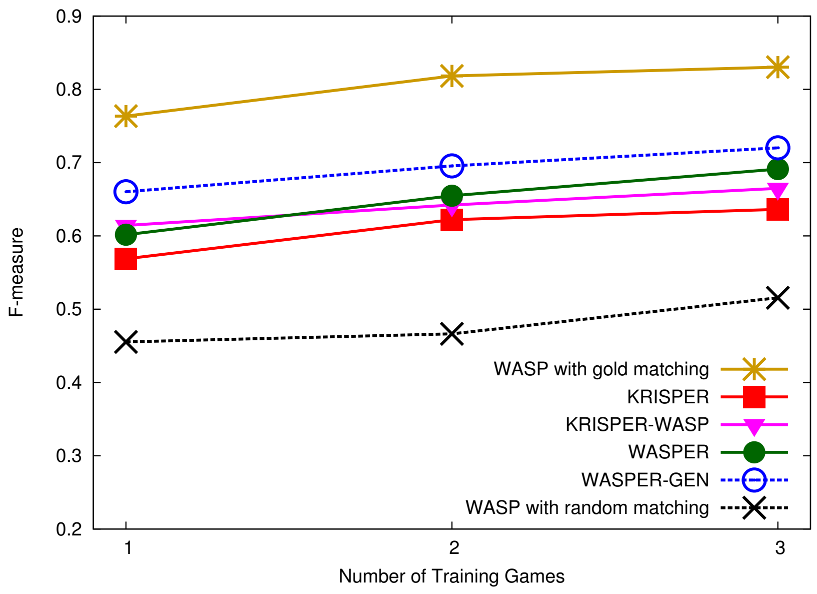


(b) Korean

Figure 3.3: Matching results on the English and Korean datasets. Wasper-Gen performs the best, outperforming the existing system Krisper on both datasets.



(a) English



(b) Korean

Figure 3.4: Semantic parsing results on the English and Korean datasets. The results largely mirrors that of the matching results with WASPER-GEN performing the best overall.

#### 3.3.4.2 Semantic Parsing

Next, we present results on the accuracy of the learned semantic parsers. Each trained system is used to parse and produce a MR for each sentence in the test set that has a correct MR in the gold-standard matching. A parse is considered correct if and only if it matches the gold standard exactly. Parsing is a fairly difficult task because there are usually multiple ways of describing the same event. For example, "player1 passes to player2" refers to the same event as "player1 kicks the ball to player2." Thus, accurate parsing requires learning all the different ways of describing an event. Synonymy is not limited to verbs. In our data, "Pink1", "PinkG" and "pink goalie" all refer to the same player. Since we are not providing the systems with any prior knowledge, they have to learn all these different ways of referring to the same entity.

The parsing results shown in Figure 3.4 generally correlate well with the matching results. Systems that did better at disambiguating the training data also did better on parsing because their supervised training data is less noisy. WASPER-GEN again does the best overall on both the English and Korean data. It is interesting to note that KRISPER did relatively well on the English data compared to its matching performance. This is because KRISP is more robust to noise than WASP (Kate & Mooney, 2006) so even though it is trained on a noisier set of data than WASPER-GEN it still produced a comparable parser.

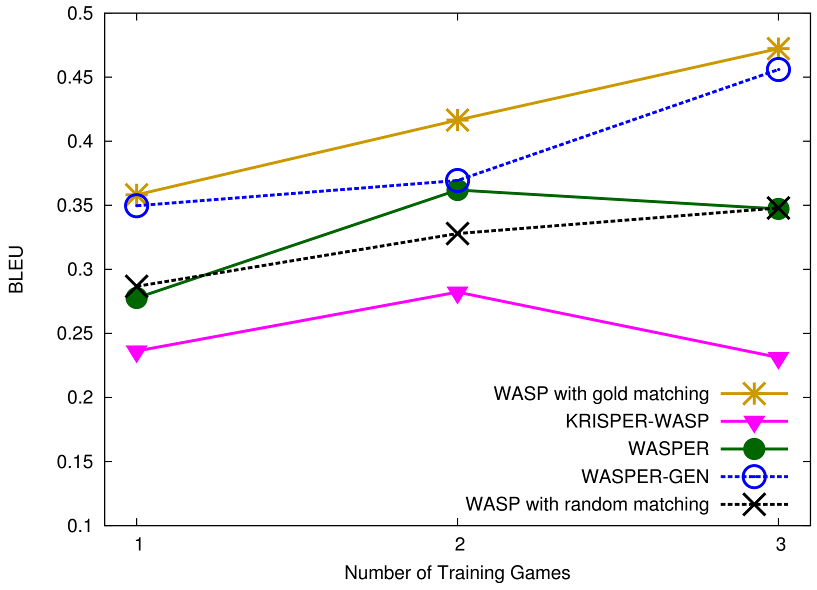
[2]KRISPER-WASP is not shown here since it uses KRISPER to disambiguate the data so it has the same performance as KRISPER on this task

#### 3.3.4.3 Generation

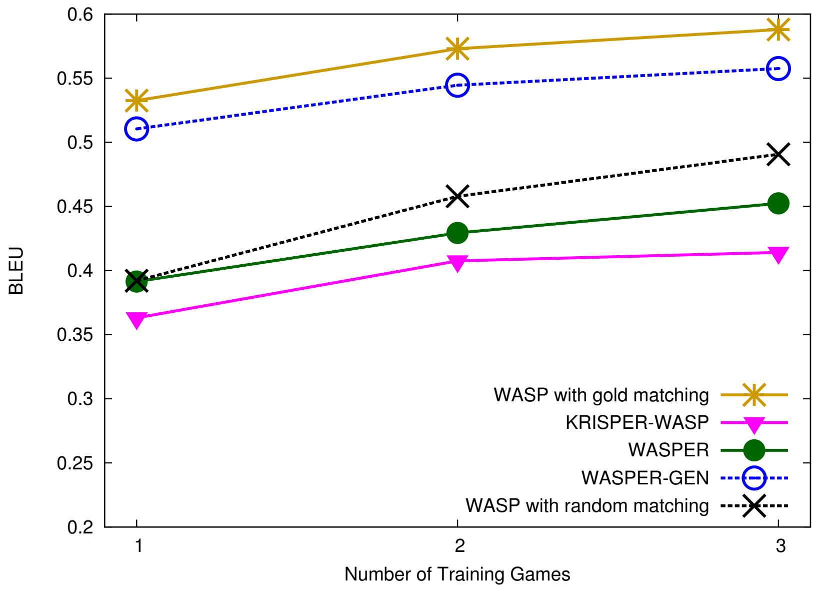
The third evaluation task is generation, or more specifically, surface realization in which a MR is transformed into a corresponding NL sentence. All of the WASP-based systems are given each MR in the test set that has a gold-standard matching NL sentence and asked to generate an NL description. The quality of the generated sentence is measured by comparing it to the gold-standard using BLEU scores.

This task is more tolerant of noise in the training data than parsing because the system only needs to learn *one way* to accurately describe an event. This property is reflected in the results, shown in Figure 3.5, where even the baseline system, WASP with random matching, does fairly well, outperforming KRISPER-WASP on both datasets and WASPER on the Korean data. As the number of event types is fairly small, only a relatively small number of correct matchings is required to perform this task well as long as each event type is associated with some correct sentence pattern more often than any other sentence pattern.

As with the other two tasks, WASPER-GEN is the best system on this task. One possible explanation of WASPER-GEN's superior performance stems from its disambiguation objective function. Systems like WASPER and KRISPER-WASP that use parsing scores attempt to learn a good translation model for each sentence pattern. On the other hand, WASPER-GEN only tries to learn a good translation model for each MR pattern. Thus, WASPER-GEN is more likely to converge on a good model as there are fewer MR patterns than



(a) English



(b) Korean

Figure 3.5: Surface realization results on the English and Korean datasets. While the relative performances of the various systems changed, WASPER-GEN is still the best system.

sentence patterns. However, it can be argued that learning good translation models for each sentence pattern will help to produce more varied commentaries, a quality that is not captured by the BLEU score. Another possible advantage for WASPER-GEN is that it uses a softer scoring function. While the probabilities of parsing from a particular sentence to a MR can be sensitive to noise in the training data, WASPER-GEN only looks at the top generated sentences for each MR. Even with noise in the data, this top generated sentence remains relatively constant. Moreover, minor variations of this sentence do not change the results dramatically since NIST score allows for partial matching.

## 3.4 Learning for Content Selection

A language generator alone is not enough to produce a sportscast. In addition to surface realization which is deciding *how* to to say something, a sportscaster must also preform *content selection* which is choosing *what* to say (McKeown, 1985).

We developed a novel method for learning which events to describe. For each event type (i.e. for each predicate like `pass`, or `goal`), the system uses the training data to estimate the probability that it is mentioned by the sportscaster. Given the gold-standard NL–MR matches, this probability is easy to estimate; however, the learner does not know the correct matching. Instead, the system must estimate the probabilities from the ambiguous training data. We compare two methods for estimating these probabilities.

Even though we do not have gold-standard matching, the systems we described in the previous section has already produced reasonable NL–MR pairs. Thus, we could treat these *inferred* matchings as gold standard matchings and estimate the probabilities directly by counting. The probability of commenting on each event type, $e_i$, is estimated as the percentage of events of type $e_i$ that have been matched to *some* NL sentence.

### 3.4.1 IGSL

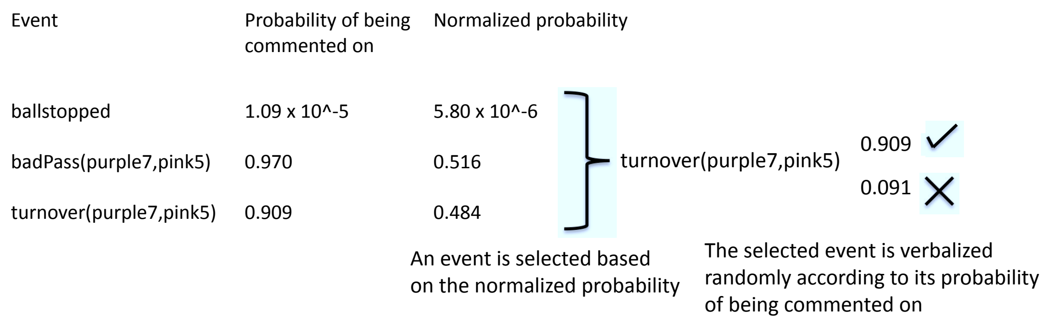
In addition to directly estimating the probabilities from the inferred matchings, we also developed a separate learning algorithm that estimates the probabilities from the ambiguous data without leveraging the outputs of the other systems. This algorithm, Iterative Generation Strategy Learning (IGSL), is a variant of EM that alternates between estimating the NL–MR matchings and computing the prior probability of each event type being commented on. Unlike our first method of using the inferred matchings, IGSL uses information about MRs not explicitly associated with any sentences in training. Algorithm 4 shows the pseudocode. The main loop alternates between two steps:

1. Calculating the expected probability of each NL–MR matching given the current model of how likely an event is commented on (line 6)
2. Update the prior probability that an event type is mentioned by a human commentator based on the matchings (line 9).

**Algorithm 4** Iterative Generation Strategy Learning (IGSL)

```
input event types $E = \{e_1, ..., e_n\}$, the number of occurrences of each event
    type $TotalCount(e_i)$ in the entire game trace, sentences $S$ and the event
    types of their associated meaning representations $Event(s)$
output probabilities of commenting on each event type $Pr(e_i)$
 1: Initialize all $Pr(e_i) = 1$
 2: repeat
 3:   for event type $e_i \in E$ do
 4:     $MatchCount = 0$
 5:     for sentence $s \in S$ do
 6:       $ProbOfMatch = \frac{\sum_{e \in Event(s) \wedge e = e_i} Pr(e)}{\sum_{e \in Event(s)} Pr(e)}$
 7:       $MatchCount = MatchCount + ProbOfMatch$
 8:     end for
 9:     $Pr(e_i) = min(\frac{MatchCount}{TotalCount(e_i)}, 1)$ {Ensure proper probabilities}
10:   end for
11: until Convergence or MAX_ITER reached
```

In the first iteration, each NL–MR match is assigned a probability inversely proportional to the amount of ambiguity associated with the sentence ($\sum_{e \in Event(s)} Pr(e) = |Event(s)|$). For example, a sentence associated with five possible MRs will assign each match a probability of $\frac{1}{5}$. The prior probability of mentioning an event type is then estimated as the average probability assigned to instances of this event type. Notice this process does not always guarantee a proper probability since a MR can be associated with multiple sentences. Thus, we limit the probability to be at most one. In the subsequent iterations, the probabilities of the NL–MR matchings are updated according to these new priors. We assign each match the prior probability of its event type normalized across all the associated MRs of the NL sentence. We then update the priors for each event type as before using the new estimated probabilities for the



| Event | Probability of being commented on | Normalized probability |
|---|---|---|
| ballstopped | 1.09 x 10^-5 | 5.80 x 10^-6 |
| badPass(purple7,pink5) | 0.970 | 0.516 |
| turnover(purple7,pink5) | 0.909 | 0.484 |

turnover(purple7,pink5)
0.909 ✓
0.091 ✗
An event is selected based on the normalized probability
The selected event is verbalized randomly according to its probability of being commented on

Figure 3.6: An example of how our content selection component works. At every timestep, we stochastically select an event from all the events occurring at that moment. We then decide whether to verbalize the selected event based on IGSL's estimated probability of it being commented upon.

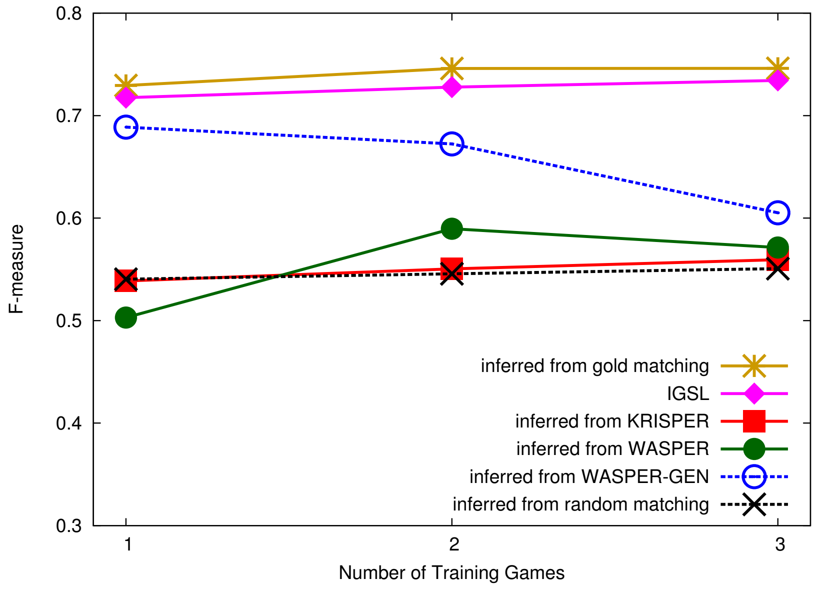
matchings. This process is repeated until the probabilities converge or a pre-specified number of iterations has occurred.

To generate a sportscast, we use the learned probabilities to determine which events to describe. For each time step, we first determine all the events that are occurring at the time. We then randomly select one based on their relative likelihood of being commented on. We do this by normalizing over the probabilities of all the events. To avoid being overly verbose, we do not want to make a comment every time something happens, especially if it is an event rarely commented on. Thus, we stochastically decide whether to comment on this selected event based on its probability. An example of this process is shown in Figure 3.6. Given the three events that were occurring, bad pass and turnover both have about the same probability of being selected. Once we selected turnover as the event to comment on, we then decide if we should generate a comment at all.

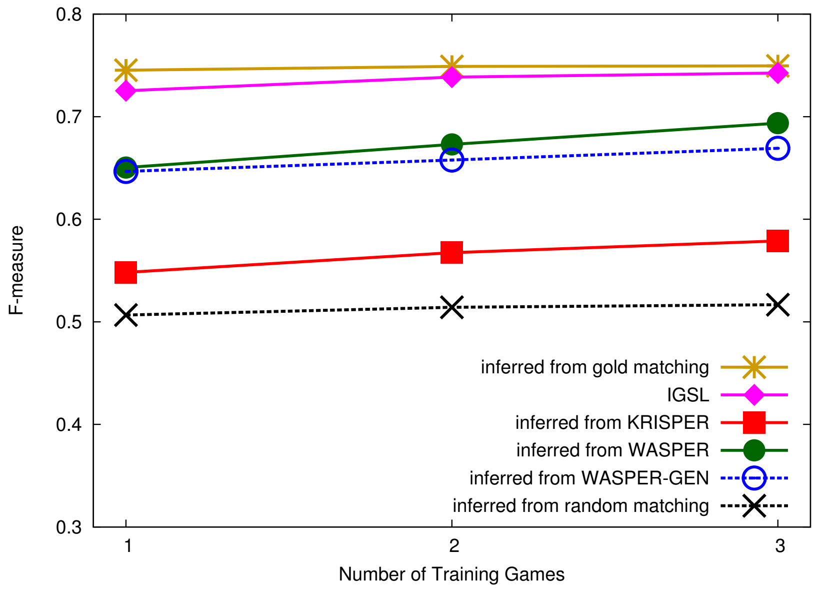
### 3.4.2 Experiments

The different methods for learning content selection were evaluated based on how often the events they describe in the test data coincide with those the humans decided to describe. For the first approach, results using the inferred matchings produced by Krisper, Wasper, and Wasper-Gen as well as the gold and random matching for establishing baselines are all presented in Figure 3.7. From the graph, it is clear that IGSL outperforms estimating the probabilities from the inferred matchings and actually performs at a level close to using the gold matching. However, it is important to note that we are limiting the potential of learning from the gold matching by using only the predicates to decide whether to talk about an event.

Next we will evaluate qualitatively the probabilities learned by IGSL and those inferred from matchings produced by Wasper-Gen. Table 3.4 shows the probabilities learned by these two methods for the five most frequent events in the English dataset. While Wasper-Gen learns fairly good probabilities in general, it does not do as well as IGSL for the most frequent events. This is because IGSL uses occurrences of events that are not associated with any possible comments in its training iterations. Rarely commented events such as *ballstopped* and *kick* often occur without any comments being uttered. Consequently, IGSL assigns low prior probabilities to them which lowers their chances of being matched to any sentences. On the other hand, Wasper-Gen does not use these priors and sometimes incorrectly matches comments to them. Thus, using the inferred matches from Wasper-Gen re-



(a) English



(b) Korean

Figure 3.7: Content selection results for our various systems. Our novel algorithm IGSL performs the best, almost on par with the upper bound which uses gold-standard matchings.

| event | # occurrences | % commented | IGSL | inferred from WASPER-GEN |
|---|---|---|---|---|
| ballstopped | 5817 | $1.72 \times 10^{-4}$ | $1.09 \times 10^{-5}$ | 0.016 |
| kick | 2122 | 0.0 33 | 0.018 | 0.117 |
| pass | 1069 | 0.999 | 0.983 | 0.800 |
| turnover | 566 | 0.214 | 0.909 | 0.353 |
| badPass | 371 | 0.429 | 0.970 | 0.493 |

Table 3.4: Top 5 most frequent events, the % of times they were commented on, and the probabilities learned by the top algorithms for the English data

sults in learning higher probabilities of commenting on these rarely commented events.

While all our methods only use the predicates of the MRs to decide whether to comment or not, they perform quite well on the data we collected. In particular, IGSL performs the best, so we use it for content selection in the rest of this chapter.

## 3.5 Human Subjective Evaluation

At best, automatic evaluation of generation is an imperfect approximation of human assessment. Moreover, automatically evaluating the quality of an entire generated sportscast is even more difficult. Consequently, we used Amazon's Mechanical Turk to collect human judgements of the produced sportscasts. Each human judge was shown three clips of simulated game video in one sitting. There were 8 video clips total. The 8 clips use 4 game segments of 4 minutes each, one from each of the four games (2001-2004 RoboCup finals). Each of the 4 game segments is commentated once by a human and once by our system. We use IGSL to determine the events to comment on and WASPER-GEN (our best performing system for surface realization) to produce the commentaries. To make the commentaries more varied, we took the top 5 outputs from WASPER-GEN and chose one stochastically weighted by their scores. The system was always trained on three games, leaving out the game from which the test segment was extracted. The video clips were accompanied by commentaries that appear both as subtitles on the screen as well as audio produced by automated text to speech systems (FreeTTS for English and TextAloud for Korean).[3] The videos are shown in random counter-balanced order to ensure no consistent bias toward segments being shown earlier or later. We asked the judges to score the sportscasts using the following metrics:

| *Score* | *Fluency* | *Semantic Correctness* | *Sportscasting Ability* |
|---|---|---|---|
| 5 | Flawless | Always | Excellent |
| 4 | Good | Usually | Good |
| 3 | Non-native | Sometimes | Average |
| 2 | Disfluent | Rarely | Bad |
| 1 | Gibberish | Never | Terrible |

Fluency and semantic correctness, or adequacy, are standard metrics in human evaluations of NL translations and generations. Fluency measures how well the commentaries are structured, including syntax and grammar. Semantic correctness indicates whether the commentaries accurately describe what

[3]Sample video clips with sound are available on the web at `http://www.cs.utexas.edu/users/ml/clamp/sportscasting/`.

is happening in the game. Finally, sportscasting ability measures the overall quality of the sportscast. This includes whether the sportscast is interesting and flows well. In addition to these metrics, we also asked them whether they thought the sportscast was produced by a human or a computer (*Human?*) as a simple form of a Turing test.

Since Mechanical Turk recruits judges over the Internet, we had to make sure that the judges were not assigning the ratings randomly. Thus, in addition to asking them to rate each video, we also asked them to count the number of goals in each video. Incorrect responses to this question caused their ratings to be discarded. This is to ensure that the judges faithfully watched the entire clip before assigning ratings. After such pruning, there were on average 36 ratings (from 40 original ratings) for each of the 8 videos for the English data. Since it was more difficult to recruit Korean judges over the Internet, we recruited them in person and collected 7 ratings on average for each video in the Korean data. Table 3.5 and 3.6 show the results for English Korean, respectively. Statistically significant results are shown in boldface.

Results are surprisingly good for the English data across all categories with the machine actually scoring higher than the human on average. However, the differences are not statistically significant based on an unpaired t-test ($p > 0.05$). Nevertheless, it is encouraging to see the machine being rated so highly. There is some variance in the human's performance since there were two different commentators. Most notably, compared to the machine, the human's performance on the 2002 final is quite good because the commentary in-

| Game Year | Commentator | *Fluency* | *Semantic Correctness* | *Sportscasting Ability* | *Human?* |
|---|---|---|---|---|---|
| 2001 | Human | 3.74 | 3.59 | 3.15 | 20.59% |
| | Machine | 3.89 | 3.81 | **3.61** | 40.00% |
| 2002 | Human | 4.13 | **4.58** | **4.03** | **42.11%** |
| | Machine | 3.97 | 3.74 | 3.29 | 11.76% |
| 2003 | Human | 3.54 | 3.73 | 2.61 | 13.51% |
| | Machine | **3.89** | **4.26** | **3.37** | 19.30% |
| 2004 | Human | 4.03 | 4.17 | 3.54 | 20.00% |
| | Machine | 4.13 | 4.38 | 4.00 | **56.25%** |
| Average | Human | 3.86 | 4.03 | 3.34 | 24.31% |
| | Machine | 3.94 | 4.03 | 3.48 | 26.76% |

Table 3.5: Human evaluation of the English sportscasts. Bold numbers indicate statistical significance between the ratings of the human and machine sportscasts.

| Game Year | Commentator | *Fluency* | *Semantic Correctness* | *Sportscasting Ability* | *Human?* |
|---|---|---|---|---|---|
| 2001 | Human | 3.75 | 4.13 | **4.00** | 50.00% |
| | Machine | 3.50 | 3.67 | 2.83 | 33.33% |
| 2002 | Human | 4.17 | **4.33** | 3.83 | 83.33% |
| | Machine | 3.25 | 3.38 | 3.13 | 50.00% |
| 2003 | Human | **3.86** | **4.29** | **4.00** | **85.71%** |
| | Machine | 2.38 | 3.25 | 2.88 | 25.00% |
| 2004 | Human | 3.00 | 3.75 | 3.25 | 37.50% |
| | Machine | 2.71 | 3.43 | 3.00 | 14.29% |
| Average | Human | **3.66** | **4.10** | **3.76** | **62.07%** |
| | Machine | 2.93 | 3.41 | 2.97 | 31.03% |

Table 3.6: Human evaluation of Korean sportscasts. Bold numbers indicate statistical significance between the ratings of the human and machine sportscasts..

cluded many details such as the position of the players, the types of passes, and comments about the overall flow of the game. On the other hand, the human's performance on the 2003 final is quite bad because the human commentator was very "mechanical" and used the same sentence pattern repeatedly. The machine performance was more even throughout although sometimes it gets lucky. For example, the machine serendipitously said "This is the beginning of an exciting match." near the start of the 2004 final clip simply because this statement was incorrectly learned to correspond to an extracted MR that is actually unrelated.

The results for Korean are not as impressive. The human beats the machine on average for all categories. However, the largest difference between the scores in any category is only 0.8. Moreover, the absolute scores indicate that the generated Korean sportscasts are at least of acceptable quality. The judges even mistakenly thought they were produced by humans one third of the time. Part of the reason for the worse performance compared to the English data is that the Korean commentaries were fairly detailed and included events that were not extracted by our limited perceptual system. Thus, the machine simply had no way of competing because it is limited to only expressing information that is present in the extracted MRs.

We also elicited comments from the human judges to get a more qualitative evaluation. Overall, the judges thought the generated commentaries were good and accurately described the actions on the field. Picking from the top 5 generated sentences also added variability to the machine-generated sportscasts that improved the results compared with earlier experiments (Chen & Mooney, 2008). However, the machine sometimes still misses significant plays such as scoring or corner kicks. This is because these plays happen much less frequently and often coincide with many other events (e.g. shooting for the goal and kickoffs co-occur with scoring). Thus, the machine has a harder time learning about these infrequent events. Another issue concerns our representation. Many people complained about long gaps in the sportscasts or lack of details. Our event detector only concentrates on ball possession and not on positions or elapsed time. Thus, a player holding onto a ball or dribbling for a long time does not produce any events detected by our simulated perceptual system. Also, a short pass in the backfield is treated exactly the same as a long pass across the field to near the goal. Finally, people were expecting more color-commentary (background information, statistics, or analysis of the game) typical of a normal sportscast to fill in the voids. This is a somewhat orthogonal issue since our goal was to build a play-by-play commentator that described events that were currently happening.

## 3.6 Using a Generative Alignment Model

After we published about our initial system (Chen & Mooney, 2008), Liang et al. (2009) developed a generative model that can be used to match natural language sentences to facts in a corresponding database to which they may refer. The generative model first selects a sequence of records, then selects a sequence of fields within the records to describe, and finally generates words

| Algorithm | English dataset | | Korean dataset | |
|---|---|---|---|---|
| | Uninitialized | Initialized | Uninitialized | Initialized |
| Liang et al. (2009) | 75.7 | | 69.4 | |
| WASPER | 59.7 | 79.3 | 72.8 | 76.6 |
| WASPER-GEN | 68.1 | 75.8 | 75.3 | 80.0 |

Table 3.7: Matching results (F1 scores) on 4-fold cross-validation for both the English and the Korean datasets. Systems run with initialization are initialized with the matchings produced by Liang et al.'s (2009) system.

| Algorithm | English dataset | | Korean dataset | |
|---|---|---|---|---|
| | Uninitialized | Initialized | Uninitialized | Initialized |
| WASP | n/a | 80.3 | n/a | 74.01 |
| WASPER | 61.84 | 79.32 | 69.12 | 75.69 |
| WASPER-GEN | 70.15 | 77.59 | 72.02 | 77.49 |

Table 3.8: Semantic parsing results (F1 scores) on 4-fold cross-validation for both the English and the Korean datasets. Systems run with initialization are initialized with the matchings produced by Liang et al.'s (2009) system.

| Algorithm | English dataset | | Korean dataset | |
|---|---|---|---|---|
| | Uninitialized | Initialized | Uninitialized | Initialized |
| WASP | n/a | 0.4580 | n/a | 0.5828 |
| WASPER | 0.3471 | 0.4599 | 0.4524 | 0.6118 |
| WASPER-GEN | 0.4560 | 0.4414 | 0.5575 | 0.6796 |

Table 3.9: Surface realization results (BLEU score) on 4-fold cross-validation for both the English and the Korean datasets. Systems run with initialization are initialized with the matchings produced by Liang et al.'s (2009) system.

based on the fields selected. Their model simultaneously segments streams of text into utterances and aligns them to their meaning representations. As one of their evaluation domains, they used our English RoboCup sportscasting data. Their method solves the matching (alignment) problem for our data, but does not address the tasks of semantic parsing or language generation. However, their generative model elegantly integrates the surface realization and content selection steps in order to find the overall most probable alignment of sentences and events. They demonstrated improved matching performance on our English data, generating more accurate NL–MR pairs than our best system, WASPER-GEN. Thus, we were curious if their results could be used to improve our own systems, which also perform semantic parsing and generation. We also ran their code on our new Korean data but that resulted in much worse matching results compared to our best system as can be seen in Table 3.7.

The simplest way of utilizing their results is to use the NL–MR pairs produced by their method as supervised data for WASP. As expected, the improved NL–MR pairs for the English data resulted in improved semantic parsers as can be seen in the results in Table 3.8. Even for the Korean dataset, training on matchings produced by their system ended up doing fairly well even though the matching performance was poor. For surface realization, using their matching only produced marginal improvement on the English dataset and a surprisingly large improvement on the Korean data as shown in Table 3.9. Overall, using the alignments produced by Liang et al.'s system resulted in good semantic parsers and language generators.

In addition to training Wasp with their alignment, we can also utilize their output as a better *starting point* for our own systems. Instead of initializing our iterative alignment methods with a model trained on *all* the ambiguous NL–MR pairs, they can be initialized with the disambiguated NL–MR pairs produced by Liang et al.'s system.

Initializing the systems in this manner almost always improved the performance on all three tasks (Tables 3.7, 3.8, and 3.9). Moreover, the results from the best systems exceeded that of simply training Wasp with their alignments in all cases except for semantic parsing on the English data. Thus, combining Liang et al.'s alignment with our disambiguation techniques seems to produce the best overall results. For the English data, Wasper with initialization performs the best on both matching and generation. It does slightly worse on the semantic parsing task compared to Wasp trained on Liang et al.'s alignment. For the Korean data, all the systems do better than just training Wasp on the alignment. Wasper-Gen with initialization performs the best on all three tasks.

Overall, initializing our systems with the alignment output of Liang et al.'s generative model improved performance as expected. Starting with a cleaner set of data led to better initial semantic parsers and language generators which led to better end results. Furthermore, by incorporating a semantic parser and a language generator, we were able to improve on the alignments produced by Liang et al.'s algorithm and achieve even better results in most cases.

## 3.7 Removing Superfluous Comments

So far, we have only discussed how to handle ambiguity in which there are multiple possible MRs for each NL sentence. During training, all our methods assume that each NL sentence matches exactly one of the potential MRs. However, some comments are *superfluous*, in the sense that they do not refer to *any* of the extracted events in the set of potential MRs. As previously shown in Tables 3.2, about one fifth of the English sentences and one tenth of the Korean sentences are superfluous in this sense.

There are many reasons for the existence of superfluous sentences. They occur naturally in language because people do not always talk about the current environment. In our domain, sportscasters often mention past events or more general information about particular teams or players. Moreover, depending on the application, the chosen MRL may not represent all of the things people talk about. For example, our RoboCup MRL cannot represent information about players who are not actively engaged with the ball. Finally, even if a sentence can be represented in the chosen MRL, errors in the perceptual system or an incorrect estimation of when an event occurred can also lead to superfluous sentences. Such perceptual errors can be alleviated to some degree by increasing the size of the window used to capture potential MRs (a 5-second window in our experiments). However, this comes at the cost of increased ambiguity because each sentence would be matched with more MRs.

To address the problem of superfluous sentences, we can eliminate the lowest-scoring NL–MR pairs (e.g. lowest parsing scores for WASPER or lowest

NIST scores for Wasper-Gen). However, in order to set the pruning threshold, we need to automatically estimate the *amount* of superfluous commentary in the absence of supervised data. Notice that while this problem looks similar to the content selection problem (estimating how likely a MR participates in a correct matching as opposed to how likely a NL sentence participates in a correct matching), the approaches used there cannot be applied here. First, we cannot use the matches inferred by the existing systems to estimate the fraction of superfluous comments since the current systems match every sentence to some MR. It is also difficult to develop an algorithm similar to IGSL due to the imbalance between NL sentences and MRs. Since there are many more MRs, there are more examples of events occurring without commentaries than vice versa.

### 3.7.1 Estimating the Superfluous Rate Using Internal Cross Validation

Since our previous techniques do not apply here, we propose a different method of estimating the superfluous rate using a form of internal (i.e. within the training set) cross-validation. First we split the training data into training and validation sets. We then train many different semantic parsers that assume different superfluous rates. Each of the semantic parser is then evaluated on the held-out validation set and the best semantic parser is chosen. We then retrain on the entire training data (training + validation) using the corresponding superfluous rate. While this algorithm can be used in conjunction with any of our systems, we chose to implement it for Krisper which trains much faster than our other systems. This makes it more tractable to train many different semantic parsers and choose the best one.

The key issue with this approach is that our held-out validation set does not contain the correct answers. Instead, they only contain potential MRs for each sentence. However, if we assume a reasonable superfluous sentence rate, then most of the time the correct MR is contained in the set of MRs associated with an NL sentence. Thus, it follows that a semantic parser that parses an NL sentence into one of the MRs associated with it is generally better than one that parses it into an MR not in the set. With this approximate method of estimating accuracy, we can evaluate the trained semantic parsers and pick the best one. The algorithm is briefly summarized in the following steps:

1. Split the training set into an *internal training set* and an *internal validation set*.

2. Train KRISPER $N$ times on the *internal training set* using $N$ different threshold values (eliminating the lowest scoring NL–MR pairs below the threshold in each retraining iteration in Algorithm 1).

3. Test the $N$ semantic parsers on the *internal validation set* and determine which parser is able to parse the largest number of sentences into one of their potential MRs.

4. Use the threshold value that produced the best parser in the previous step to train a final parser on the complete original training set.

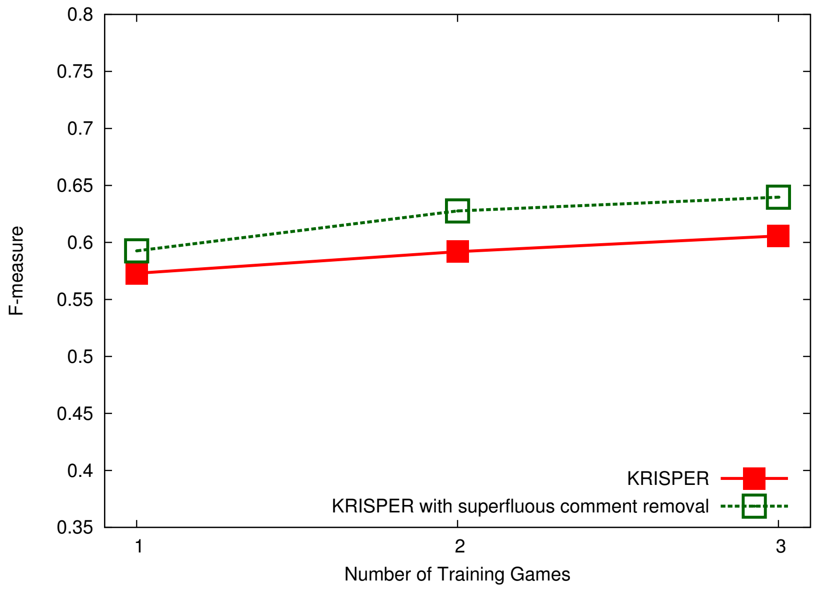
### 3.7.2 Experiments

We evaluated the effect of removing superfluous sentences on all three tasks: matching, parsing, and generation. We present results for both KRISPER and KRISPER-WASP. For matching, we only show results for KRISPER because it is responsible for disambiguating the training data for both systems (so KRISPER-WASP's results are the same). For generation, we only show results for KRISPER-WASP, since KRISPER cannot perform generation.

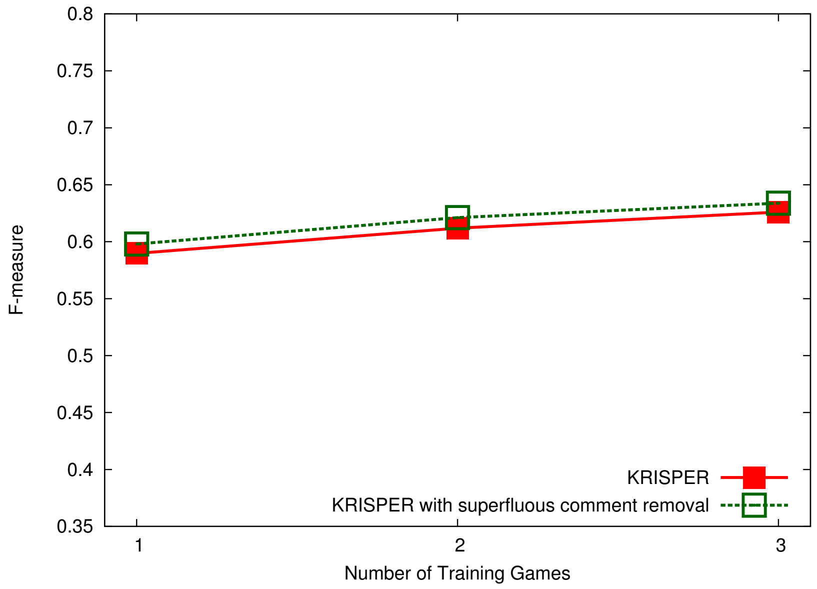
The matching results shown in Figure 3.8 demonstrate that removing superfluous sentences does improve the performance for both the English and Korean data, although the difference is small in absolute terms.

The parsing results shown in Figure 3.9 indicate that removing superfluous sentences usually improves the accuracy of both KRISPER and KRISPER-WASP marginally. As we have observed many times, the parsing results are consistent with the matching results.

Finally, the surface realization results shown in Figure 3.10 suggest that removing superfluous comments actually decreases performance somewhat. One potential explanation is that generation is less sensitive to noisy training data. While removing superfluous comments improves the purity of the training data, it also removes potentially useful examples. Consequently, the system does not learn how to generate sentences that were removed from the data. Overall, for generation, the advantage of having cleaner disambiguated training data is likely outweighed by the loss of data.

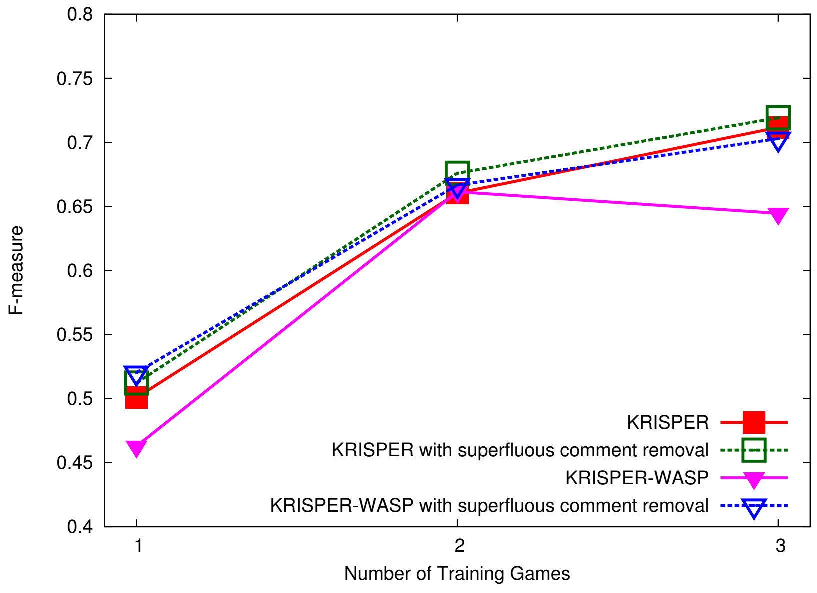


(a) English

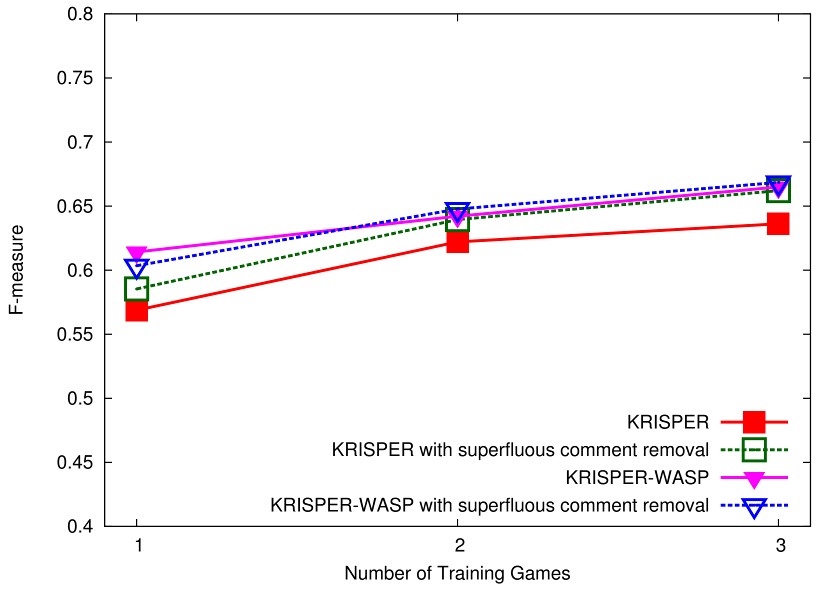


(b) Korean

Figure 3.8: Matching results comparing the effects of removing superfluous comments.

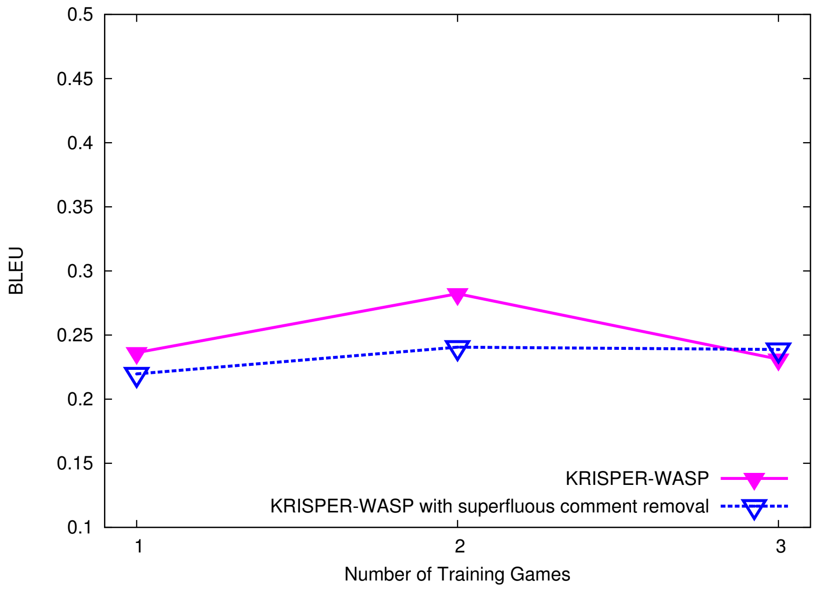


(a) English

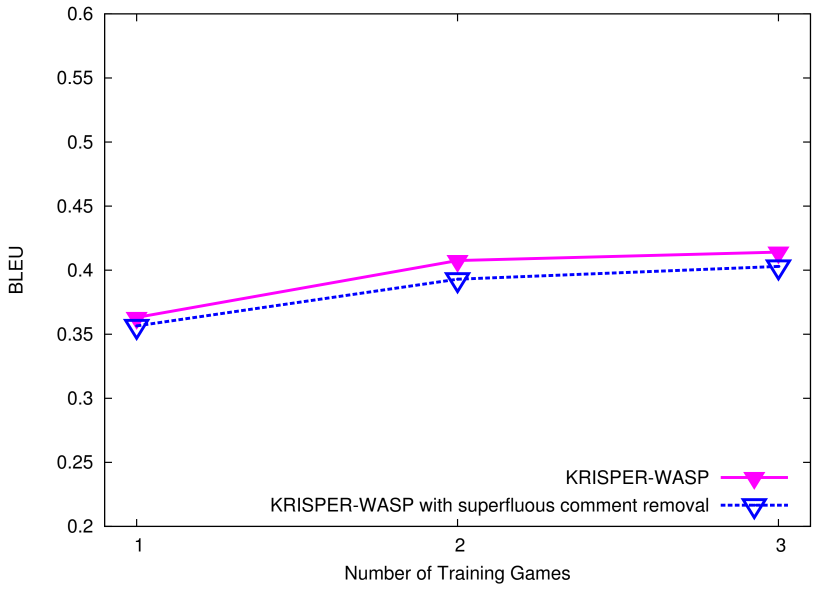


(b) Korean

Figure 3.9: Semantic parsing results are improved marginally after superfluous comment removal.



(a) English



(b) Korean

Figure 3.10: Surface realization performance decreases after removing superfluous comments.

## 3.8 Discussion and Possible Extensions

The systems presented so far are limited in the scope of problems that they can solve. While our framework is very flexible in that we can use practically any scoring function to determine the best NL–MR pairs, it is limited in its scalability. In particular, we need to enumerate all the possible MRs for a NL sentence and score each alignment. Thus, our computation time grows at least linearly with the amount of ambiguity. Moreover, the initial iteration of the algorithm (assuming we are not initializing the system with an initial matching) requires training a translation model with all the possible pairings. This is typically a very expensive operation where the computation complexity increases more than just linearly with the number of pairings. This limits the framework to only deal with ambiguous training data where there are only a handful of choices for each sentence.

In addition to the pure computational issue, there is a separate issue in that the different MRs are considered separately. This precludes any interaction between the different MRs. For example, two MRs could have non-trivial overlapping components, or one MR could be temporally related to another MR. For example, a pass is preceded by a kick, and a bad pass is followed by a turnover. A more natural way to represent these kinds of relationships is to use a graph to represent not only the entities and events but also the relationships between them. We explore both the computational issues and the relationships between entities further in our navigation domain where the space of potential MRs is represented as a connected graph rather than a list of independent events.

## 3.9 Chapter Summary

We have presented an end-to-end system that learns to generate natural-language sportscasts for simulated RoboCup soccer games by training on sample human commentaries paired with automatically extracted game events. By learning to semantically interpret and generate language without explicitly annotated training data, we have demonstrated that a system can learn language by simply observing linguistic descriptions of ongoing events. We demonstrated the system's language independence by successfully training it to produce sportscasts in both English and Korean. Finally, our system also learns a model for content selection from the ambiguous training data by estimating the probability that each event type evokes human commentary.

Dealing with the ambiguous supervision inherent in the training environment is a critical issue in learning language from perceptual context. We have evaluated various methods for disambiguating the training data in order to learn semantic parsers and language generators. Using a generation evaluation metric as the criterion for selecting the best NL–MR pairs produced better results than using semantic parsing scores when the initial training data is very noisy. We also demonstrated that our system can be initialized with alignments produced by a different system to achieve better results than either system alone. Finally, experimental evaluation verified that the overall system learns to accurately parse and generate comments and produce sportscasts that are competitive with those produced by humans.

However, the framework presented so far is also limited in its ability to scale to larger and more complex scenarios. In particular, it requires enumerating each of the possible NL–MR pairings to select the best ones. We address this problem in the next chapter in which each NL sentence can be aligned to an exponential number of possible MRs. Thus, we have to perform the alignment without explicitly enumerating all the possible choices. Another issue is representing the relationships between the different MRs. While thus far we have presented a scenario where each potential MR is independent from one another, for many applications the MRs are related (e.g. temporally, or hierarchically where one concept encompasses the other). We also address this issue in our navigation domain where the MRs are connected by edges to represent temporal relationships or argument structures.

# Chapter 4

# Learning from Ambiguously Supervised Relational Data

In this chapter we will look at a more difficult ambiguous supervision setting in which the number of possible alignments is exponential. In particular, we will examine this problem in the context of building a (virtual) mobile robot that can following natural language navigation instructions. In addition to the increased amount of ambiguity in the training data, we also take into account the relationships between the different semantic entities. We represent the MRs as connected graphs where the edges represent either temporal relationships or argument structures.

The goal of our navigation system is to transform natural language instructions into executable formal plans. Given no prior linguistic knowledge, the system learns by simply observing how humans follow navigation instructions. The training data consists of textual navigation instructions and the corresponding sequences of primitive actions (forward, turn left, turn right) recorded from the human followers. We construct the space of possible navigation plans from these action sequences and represent it as a connected graph. We then align each instruction to a plan in this space (a connected subgraph).

Since there are an exponential number of subgraphs, our previous approach of enumerating all possible alignments and scoring each one becomes intractable for moderately large graphs. Instead, we first learn a lexicon of words and short phrases and use it to prune the nodes in the graph. We then train Krisp, a supervised semantic parser learner, on the resulting refined graphs.

We evaluate the system on data collected from three complex virtual indoor environments, each containing several objects and landmarks. The system is able to find good alignments and execute a majority of the single-sentence instructions. However, its performance on longer instructions is lower and overall it does worse than humans and a previously hand-built system. Given that one advantage of a learning system is that it can adapt to new domains or languages, we also translated all the English instructions into Mandarin Chinese. The system performed similarly on the Chinese data as on the English data, again demonstrating the generality of our systems.

## 4.1 Overview

We have already looked at one method of dealing with ambiguous supervision in Chapter 3. By using an EM-like algorithm, we alternated between training a translation model and selecting the best NL–MR pairs by scoring all possible alignments for each sentence. The complexity of the algorithm thus grows at least linearly with the number of potential MRs for each sentence. In practice, the complexity is usually even worse because we have to train our initial translation model with all the possible pairings.

The amount of ambiguity in the supervision partly depends on the MRL. For example, in the sportscasting task we have concentrated on generating events related to the player in control of the ball. Had we generated events for every player as they moved around the field (e.g. pink4 is running down the sideline), the amount of ambiguity would likely grow by an order-of-magnitude. In addition to the issue of the range of events to include in the MRL, the granularity of the MRL also affects the amount of ambiguity. For example, we represented a passing event with just two arguments: the passer and the receiver. We did not include any information about the type of the pass (e.g. a quick pass or a pass across the field) or the location of the pass (e.g. a pass backward or a pass down to near the corner). If we included such information we would then need to allow partial matchings since not all NL sentences would contain every detail. This leads to an exponential number of possible alignments as we need to consider all possible combinations of whether to include a particular argument or not.

One way to address the combinatorial problem is to make independence assumptions. For example, we could use a generative model to first select the MRs, then select the arguments of each MR to describe, and finally generate the actual words (Liang et al., 2009). However, such an approach ignores the relationships between different MRs and also between the different arguments of a MR. Thus, we take a different approach in solving this problem. Instead of trying to initially map entire sentences to their representations, we first build a lexicon of words and short phrases. We encode the relationships between the semantic entities as edges in a graph. The meanings of each word or short phrase thus correspond to connected subgraphs. Using this learned lexicon, we can then refine the potential MRs by removing parts of the graphs that are irrelevant.

To study this more complicated scenario of learning from ambiguously supervised relational data where the MRs are related rather than independent records, we look at the task of interpreting navigation instructions (MacMahon et al., 2006; Shimizu & Haas, 2009; Matuszek, Fox, & Koscher, 2010; Kollar, Tellex, Roy, & Roy, 2010; Vogel & Jurafsky, 2010). An important application of natural language processing is understanding 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 learning how to map natural-language instructions into actions that can be performed by a computer (Branavan et al., 2009; Lau, Drews, & Nichols, 2009).

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 directions are 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 done by linguists (Klein, 1982; Wunderlich & Reinelt, 1982). While this domain is restricted, there is considerable variation in how different people describe

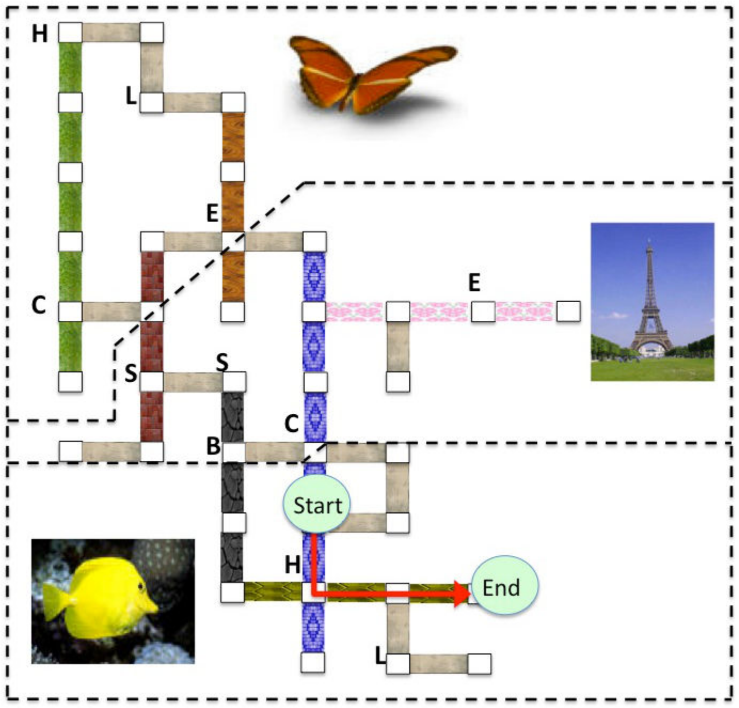


Figure 4.1: This is an example of a route in our virtual world named Jelly. The world consists of interconnecting hallways with varying floor tiles and paintings on the wall (butterfly, fish, or Eiffel Tower.) Letters indicate objects (e.g. 'C' is a chair and 'H' is a hatrack) at an intersection.

the same route. Below are some examples from our test corpus of instructions given for the route shown in Figure 4.1:

> "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 [sic] 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 details 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 chapter 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 longer instructions where error recovery is necessary.

The rest of the chapter is organized as follows. We first formally define our learning problem and the virtual environment we use to test our navigation system in Section 4.2. We then present the basic framework of our system along with some experimental results in Section 4.3. We then describe an alternate online lexicon learning algorithm that can scale to larger data in Section 4.4. Additional experimental results are presented in Section 4.5 and 4.6, containing experiments on modifying the parameters of our system and testing with different folds and translations of the data, respectively. Section 4.7 describes our effort in collecting additional training data using Mechanical Turk. Finally, we discuss some potential extensions to the framework in Section 4.8 and summarize the chapter in Section 4.9.

## 4.2 Navigation Task and Data

The goal of the navigation task is to build a system that can understand free-form natural-language instructions and follow them to move to the desired destination. We again approach the problem assuming no prior linguistic knowledge: syntactic, semantic, or lexical. Consequently, we have to learn the meanings of all the 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 world including the patterns of the floors and walls as well as the locations of the 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 4.1 showed, several different plans can be used to navigate the same route. In other words, there 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 $p_i$. This scenario is similar to the problem of building semantic parsers from question and answer pairs (Clarke, Goldwasser, Chang, & Roth, 2010; Liang et al., 2011). The semantic representations are not observed, but must be inferred from their effects on the world (e.g. actions performed following a navigation plan or retrieved answer for a question.)

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 (Grid, L, and Jelly) consisting of interconnecting hallways. An overview map of one of the worlds, Jelly, is shown in Figure 4.1. 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 are also furnitures placed at various intersections (hatrack, lamp, chair, sofa, barstool, and easel). The three worlds contain the same elements but are arranged in different configurations. Grid has the most compact design, resulting in a grid-like world. L is slightly more spread out while Jelly has the sparest layout. Each world also has seven chosen positions labeled 1 thorough 7. Details of the three worlds can be found in Appendix B.

MacMahon et al. (2006) 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 and recall the routes from memory. 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. Thus, each training example for our system consists of only a single sentence paired with a sequence of actions taken in response to this instruction. All the actions are discrete and there are only three action types in total: turning left, turning right, and moving from one intersection to another. In addition to collecting

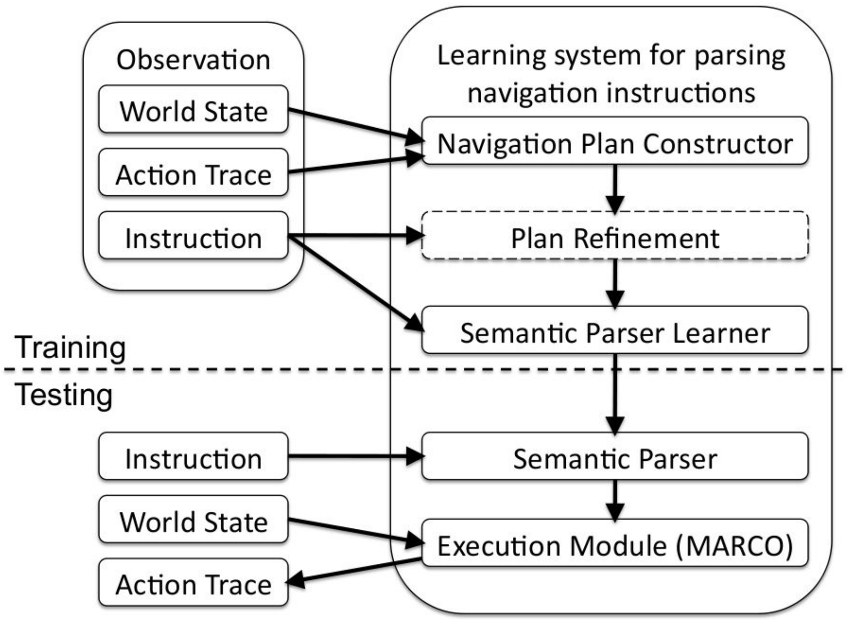


Figure 4.2: An overview of our system

instruction and follower data, MacMahon et al. (2006) also developed a system called Marco that can follow natural language navigation instructions. We utilize part of Marco in our own navigation system.

## 4.3 System Description

Figure 4.2 shows the general framework of our system. 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.

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. Specifically, we use KRISP to train our semantic parsers and MARCO's execution module (MacMahon et al., 2006) for executing the navigation plans in our test environments. We will now examine each component in more details.

### 4.3.1 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 actions 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 4.3.

We define a formal language called Steering Action Instruction Lan-

```
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 4.3: Examples of automatically generated plans.

guage (SAIL) that can represent both the *basic plans* and the *landmarks plans*. Details about SAIL and the CFG for the language can be found in Appendix B.

### 4.3.2 Plan Refinement

While *landmarks plans* capture more of the meaning of the instructions, they usually also include a lot of superfluous information. Thus, we need to determine what information are actually included in the NL instructions. This turns into the combinatorial alignment problem we mentioned in Section 4.1. Moreover, we want to maintain the temporal relationships between the actions as the order of the actions is critical for correct navigation.

Instead of trying to determine the meaning of the entire training sentence as we did in Chapter 3, we first learn a lexicon of words and short phrases. The learned lexicon is then used to try to identify and remove the extraneous details in the *landmarks plan*.

#### 4.3.2.1 Learning a Lexicon

We build a semantic lexicon by finding the common parts of the formal representations associated with different occurrences of the same word or phrase (Siskind, 1996). More specifically, we represent the navigation plans as graphs and compute their common parts by finding an intersection between them (Thompson & Mooney, 2003). Here we use the term intersection to mean a maximal common subgraph such that there does not exist a common subgraph that contains more nodes. In general, there might be multiple possible intersections between two graphs. For our intersection operation, we bias toward finding large connected components. We greedily remove the largest common connected 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 4.4. Given two navigation plans, we first find the largest common connected subgraph. In this case, this is a graph with 4 nodes and shown in the bottom left corner of the figure. We then remove this subgraph from both of the navigation plans, and find the next largest common connected subgraph which is shown in the bottom right corner of the figure. After this subgraph is removed, there are no more common nodes between the two navigation plans so the intersection operation is complete. The intersection thus consists of the two subgraphs shown.

Using the intersections to help us find candidate meanings, we build lexical entries for all $n$-grams in the training data. Pseudo-code for our Graph

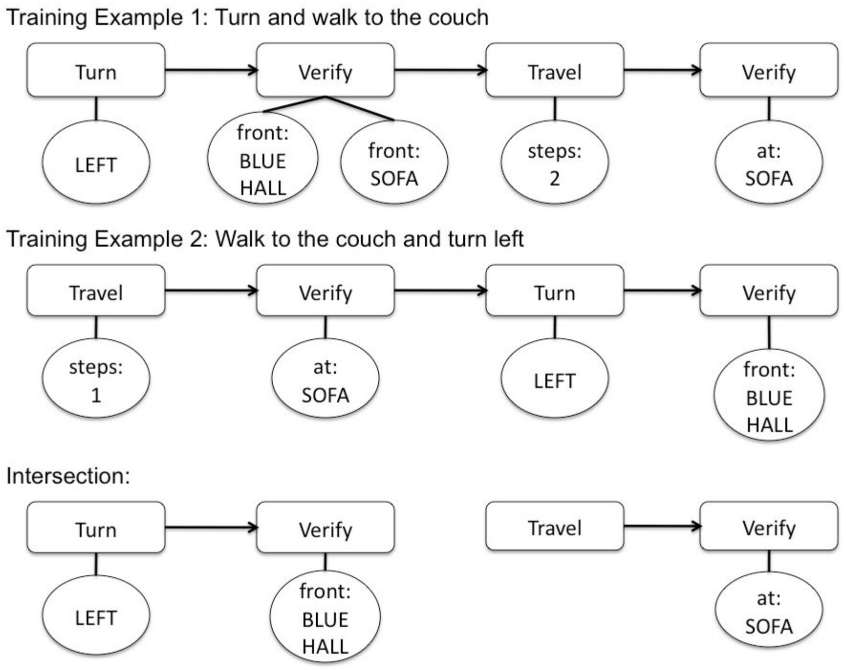


Figure 4.4: Example of computing the intersections of two graph representations of navigation plans.

Intersection Lexicon Learning (GILL) algorithm is shown in Algorithm 5. 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 each connected component of the intersections to $meanings(w)$ until further intersections do not produce any new entries. 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 $w$. Unless otherwise specified, we consider only unigrams and bigrams, and use threshold $t = 0.4$ and

**Algorithm 5** GRAPH INTERSECTION LEXICON LEARNING (GILL)

```
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
 1: main
 2:   for n-gram $w$ that appears in $\mathbf{e} = (e_1, \ldots, e_n)$ do
 3:     for instruction $e_i$ that contains $w$ do
 4:       Add navigation plan $p_i$ to $meanings(w)$
 5:     end for
 6:     repeat
 7:       for every pair of meanings in $meanings(w)$ do
 8:         Add intersections of the pair to $meanings(w)$
 9:       end for
10:       Keep $k$ highest-scoring entries of $meanings(w)$
11:     until $meanings(w)$ converges
12:     Add entries of $meanings(w)$ with scores higher than threshold $t$ to
        Lexicon
13:   end for
14: end main
```

maximum meaning set size $k = 100$.[1]

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

[1]We used $k = 1000$ in our previously published paper (Chen & Mooney, 2011) which required a lot more computational time. Consequently, there are some minor differences between the results presented here than ones previously published.

$w$ is monosemous.[2] However, the reverse is not true since an object or action may often be referred to by 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.

#### 4.3.2.2 Refining Navigation Plans Using the Lexicon

One we have built a lexicon, we will use it to help us 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 we are quite certain that they are not mentioned in the instructions. We do this by removing nodes that do not correspond to the meaning of any of the words in the instructions. Pseudo-code for our algorithm for refining the navigation plans is shown in Algorithm 6.

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 all occurrences of $w$ from $e_i$ and mark all occurrences of $g$ in $p_i$, ignoring any

[2]Notice that the actual magnitude of $p(g|w)$ matters and not just the ratio between $p(g|w)$ and $p(g|\neg w)$. Using odds ratios as the scoring function did not work as well.

**Algorithm 6** PLAN REFINEMENT

```
input A navigation instruction $e$, the corresponding navigation plan $p$ and a
    lexicon $L$
output A refined navigation plan $p'$
 1: main
 2:   for highest to lowest scoring entry $(w, g)$ in $L$ do
 3:     if $w$ in $e$ and $g$ in $p$ then
 4:       remove all occurrences of $w$ from $e$
 5:       mark all occurrences of $g$ in $p$
 6:       if $e$ is empty then
 7:         break
 8:       end if
 9:     end if
10:   end for
11:   remove any node in $p$ that was not marked and return the remaining
      graph $p'$
12: end main
```

redundant markings. This process is repeated until no words remain in $e_i$ or the entire lexicon has been exhausted. Finally, we remove all nodes in $p_i$ that were not marked and the remaining graph becomes the new *refined plan* $p'_i$. If after removing the unmarked nodes the remaining graph becomes disconnected, we add edges between the actions so they preserve the original ordering. If there are any orphan arguments that are not connected to any actions, we simply drop them from the graph.

An example of the plan refinement process is shown in Figure 4.5. Given the instruction “walk to the couch and turn left” and the associated landmarks plan, we look for lexical entries that map words in the instruction to subgraphs of the landmarks plan. In this example, we select the lexical entries for “turn left” and “walk to the couch” based on their scores. Removing these n-grams

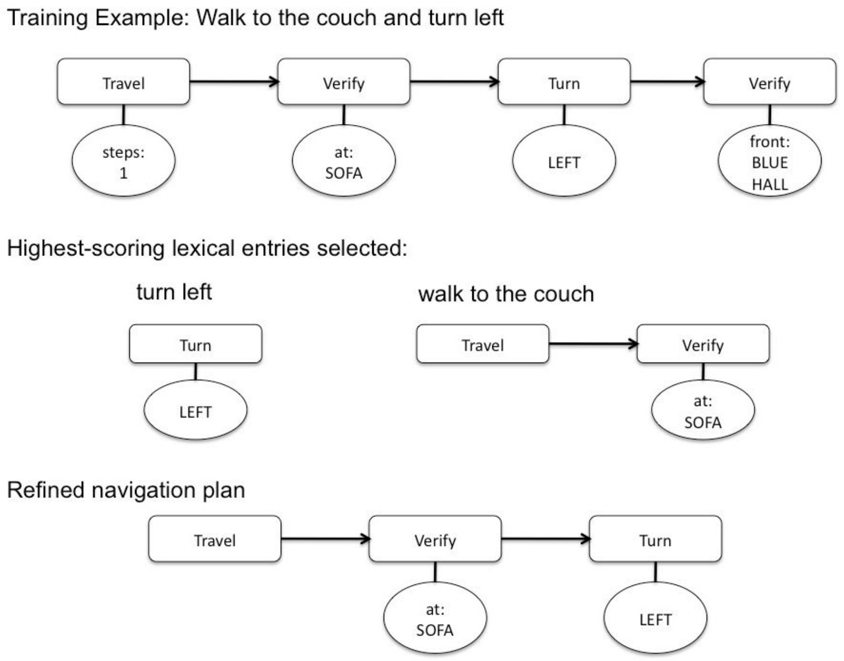


Figure 4.5: Example of refining the landmarks plan using the lexicon. Only nodes that correspond to the lexical entries selected are kept.

from the instruction leaves us with only the word "and". After failing to find any lexical entry ("and", $g$) where $g$ appears in the given graph, we remove all the nodes in the graph that do not correspond to the lexical entries for "turn left" and "walk to the couch". The bottom row shows the result of the resulting refined landmarks plan $p'$.

### 4.3.3 Learning a Semantic Parser

Once we obtain the supervised data in the form of $(e_i, p_i)$, we can train a semantic parser that will be responsible for transforming novel instructions $e_j$ into navigation plans $p_j$ (i.e. transform *turn to face the sofa* into Turn(), Verify(front: SOFA).) Since the plans inferred by the system are not always completely accurate representations of the instructions, we chose to use KRISP which has been shown to be particularly robust to noisy training data (Kate & Mooney, 2006). Nevertheless, other general-purpose supervised semantic-parser learners (Zettlemoyer & Collins, 2005; Wong & Mooney, 2006; Lu et al., 2008) could also be used.

### 4.3.4 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 et al., 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 in SAIL into a *compound action specification.* Since SAIL represents only a subset of the concepts expressible using compound action specifications, this transformation is deterministic and straightforward. The resulting compound action specification is then given to MARCO's executor to be carried out in the virtual worlds.

### 4.3.5 Experimental Evaluations

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 the three virtual worlds Grid, L, and Jelly. The instructions were written by six instructors (3 male and 3 female) for 126 unique starting and ending position pairs spread evenly across the three worlds. A separate group of 36 subjects (21 male, 15 female) followed these instructions and had their actions (turn left, turn right, or walk forward from one intersection to the next) recorded. Each instruction had between 1 to 15 human followers.

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

|  | 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 4.1: 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.

segmented data can be seen in Table 4.1. We use the single-sentence version of the corpus for training and both versions for testing.

#### 4.3.5.1 Inferring 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

|  | 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 4.2: Partial parse accuracy of how well the inferred navigation plans match gold-standard annotations.

other such that precision is maximized. Recall is computed similarly with the roles of the inferred and gold-standard plans swapped. We also compute the F1 score, the harmonic mean of precision and recall.

The results are shown in Table 4.2. 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 pairings of maps. 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.

#### 4.3.5.2 Building a Semantic Parser

Next, we evaluated the performance of the semantic parsers trained on these inferred plans. We also compared to semantic parsers trained on gold-

|  | Precision | Recall | F1 |
|---|---|---|---|
| Basic plans | 86.68 | 48.62 | 62.21 |
| Landmarks plans | 50.40 | 31.10 | 38.39 |
| Refined landmarks plans | 90.16 | 55.41 | 68.59 |
| Gold-standard plans | 88.24 | 71.70 | 79.11 |

Table 4.3: Partial parse accuracy of how well the semantic parsers trained on the different navigation plans performed on held-out test data.

standard plans as an upper baseline. We used a leave-one-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 the gold-standard plans using partial correctness as before. The results are shown in Table 4.3. 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.

#### 4.3.5.3 Executing Navigation Plans

Next, we tested our end-to-end system by executing the parsed navigation plans to see if they lead to the desired destinations. We evaluated on both the single-sentence and the original (multi-sentence) versions of the

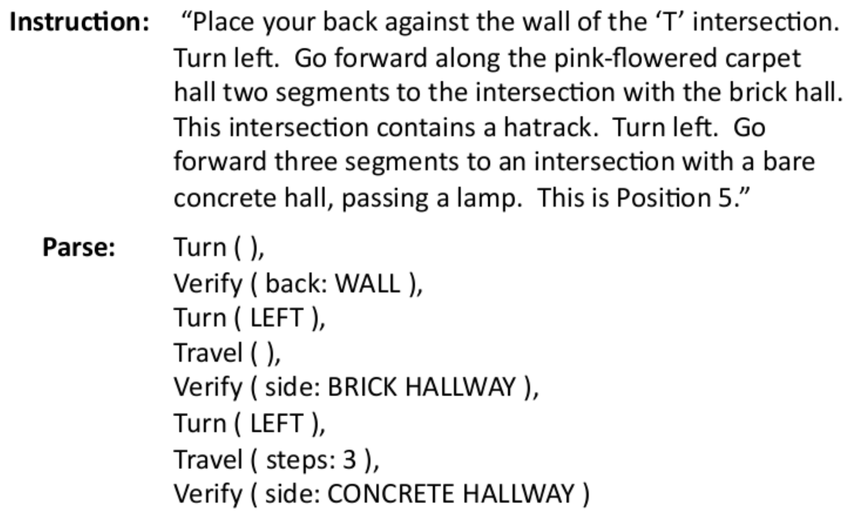
| | Single-sentence | Complete |
|---|---|---|
| Simple generative model | 11.08% | 2.15% |
| Learning from basic plans | 56.99% | 13.99% |
| Learning from landmarks plans | 21.95% | 2.66% |
| Learning from refined landmarks plans | 54.18% | 16.19% |
| Learning from gold-standard plans | 58.29% | 26.15% |
| MARCO | 77.87% | 55.69% |
| Human followers | N/A | 69.64% |

Table 4.4: Experimental results comparing different versions of our learning system and several baselines on following both the single-sentence and the complete instructions. The numbers are the percentages of trials that resulted in reaching the correct destinations.

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 original, 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 4.4.

In addition to evaluating the trained semantic parsers, we also compared to several lower and upper baselines. We 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. It estimates from the data the probability distribution over the number of actions, the probability of each action type occurring, and the probability of each argument occurring given the action type. 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 to the performance of the semantic parser trained on the gold-standard plans. This represents what could be achieved if we were able to solve the ambiguous supervision problem perfectly and produce clean supervised data to train on. Both the *basic plans* and *refined landmarks plans* approach this performance on the simpler, single-sentence task. To better understand what could be achieved by 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 (see Section 4.6.2) and environments with different objects and landmarks. On the other hand, MARCO was fully manually-engineered for this environment and hand-tuned on this specific data to achieve the best performance. As expected, human followers performed the


**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 4.6: 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.

best, although even they were still only able to complete 70% of the tasks,[3] 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. Between the systems trained on the *basic plans* and the *refined landmarks plans*, *basic plans* performed slightly better on the single-sentence task and *refined landmarks plans* performed 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.

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

The additional benefit of modeling the 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 4.3). A sample parse for this system is shown in Figure 4.6. While the plan is not a perfect representation of the instruction, it contains sufficient details to complete the task successfully in all trials.

## 4.4 Online Lexicon Learning

Central to our navigation system is the GILL algorithm described in Section 4.3.2.1. The lexicon it learns is our primary tool for solving the ambiguous supervision problem. We refine the navigation plans by removing parts of the plans that do not correspond to any of the words or phrases in the instructions according to the lexicon. Consequently, learning a good lexicon is vital to the success of our overall system.

One issue we have not considered so far is the scalability of our systems. Part of the motivation for studying language learning from ambiguous supervision is the abundance of training data potentially available since no semantic annotations are required. However, to make use of such data, our learning system must be computationally efficient to realistically scale to large datasets. One problem with GILL is that the intersection operation can be quite expensive. It requires finding the largest common connected subgraph between two graphs which is a time-consuming step. Moreover, the algorithm requires computing the intersections between every pair of potential meanings. While we can limit the beam size ($k$) to reduce the computation time, doing so could also hurt the quality of the lexicon learned.

In this section, we present another lexicon learning algorithm that is much faster than GILL and could be used in an online setting. The main insight is that most words or short phrases correspond to small graphs. Thus, we could concentrate our attention on only candidate meanings that are less than a certain size. Using this constraint, we could generate all the potential small connected subgraphs for each navigation plan in the training examples and discard the original graph. We will call this new algorithm the Subgraph Generation Online Lexicon Learning (SGOLL) algorithm. Pseudo-code for SGOLL is shown in Algorithm 7.

As we encounter each new training example that consists of a navigation instruction and a corresponding navigation plan, we will update the co-occurrence count between all n-grams $w$ that appear in the instruction and all connected subgraphs $g$ of the navigation plan with size less than or equal to $m$. We will also update the counts of how many examples we have encountered and counts of the n-grams $w$ and subgraphs $g$. At any given time, we can compute a lexicon using these various counts. Specifically, for each n-gram $w$, we will look at all the subgraphs $g$ that co-occurred with it, and compute a score for the pair $(w, g)$. If the score is higher than the threshold $t$, we will add the entry $(w, g)$ to our lexicon. We use the same scoring function as before, which

**Algorithm 7** SUBGRAPH GENERATION ONLINE LEXICON LEARNING (SGOLL)

```
input  A sequence of navigation instructions and the corresponding navigation
    plans $(e_1, p_1), \ldots, (e_n, p_n)$
output  *Lexicon*, a set of phrase-meaning pairs
 1: main
 2:   for training example $(e_i, p_i)$ do
 3:     Update($(e_i, p_i)$)
 4:   end for
 5:   OutputLexicon()
 6: end main
 7:
 8: function Update(training example $(e_i, p_i)$)
 9:   for n-gram $w$ that appears in $e_i$ do
10:     for connected subgraph $g$ of $p_i$ such that the size of $g$ is less than or
        equal to $m$ do
11:       Increase the co-occurrence count of $g$ and $w$ by 1
12:     end for
13:   end for
14:   Increase the count of examples, each n-gram $w$ and each subgraph $g$
15: end function
16:
17:
18: function OutputLexicon()
19:   for n-gram $w$ that has been observed do
20:     for subgraph $g$ that has co-occurred with $w$ do
21:       if score($w$, $g$) $>$ threshold $t$ then
22:         add $(w, g)$ to *Lexicon*
23:       end if
24:     end for
25:   end for
26: end function
```

can be computed efficiently using the counts we keep. We again consider only unigrams and bigrams, and use threshold $t = 0.4$, and maximum subgraph size $m = 3$ unless otherwise specified.

It should be noted that SGOLL can also become computationally intractable if the sizes of the navigations plans are large or if we set the maximum subgraph size $m$ to a large number. Moreover, the memory requirement can be quite high if there are many different subgraphs $g$ associated with each n-gram $w$. To deal with such scalability issues, we could use beam-search as we did in GILL, and only keep the top $k$ candidates associated with each $w$. Another important step is to define canonical orderings of the nodes in the graphs. This allows us to determine if two graphs are identical in constant time and also lets us use a hash table to quickly update the co-occurrence and subgraph counts. Thus, even given a large number of subgraphs for each training example, each subgraph can be processed very quickly. Finally, this algorithm readily lends itself to being parallelized. Each processor would get a fraction of the training data and compute the counts individually. Then the counts can be merged together at the end to produce the final lexicon.

### 4.4.1 Experiments

While computational efficiency is the main goal for designing the SGOLL algorithm, we must first verify that it can solve the ambiguous supervision problem for the navigation task. We measure its performance on three tasks as before. First, we compute the partial precision and recall of the refined

|  | Precision | Recall | F1 |
|---|---|---|---|
| GILL | 78.54 | 78.10 | 78.32 |
| SGOLL | 82.91 | 71.28 | 76.65 |

Table 4.5: Partial parse accuracy of how well SGOLL can infer the gold-standard navigation plans.

plans generated using the lexicon learned by SGOLL. Then we measure the performance of the semantic parsers trained on these refined plans. Finally, we evaluate on the end-to-end navigation task. We include the performance of GILL as reported in Section 4.3.5 for easy comparisons. All the results reported are from using the lexicons learned by GILL and SGOLL to refine the landmarks plans.

First, we examine the quality of the refined navigation plans produced using SGOLL's lexicon. The precision, recall, and F1 of these plans compared to gold-standard annotations are shown in Table 4.5. Compared to GILL, the plans produced by SGOLL has higher precision and lower recall. This is due to the fact that SGOLL explicitly limits the size of the subgraphs in the lexicon (in this case, a maximum of size 3). On the other hand, graph intersections often produce large graphs because there are usually a lot of similarities between the different navigation plans. Consequently, GILL is more likely to mark more of the original landmarks plan and retain more of the nodes.

Next we look at the performance of the semantic parsers trained on the navigation plans produced by SGOLL. The results are shown in Table 4.6. We see that the numbers are almost identical, an encouraging sign that SGOLL

|  | Precision | Recall | F1 |
|---|---|---|---|
| GILL | 90.16 | 55.41 | 68.59 |
| SGOLL | 90.04 | 55.06 | 68.31 |

Table 4.6: Partial parse accuracy of the semantic parsers trained on navigation plans produced by SGOLL.

|  | Single-sentence | Complete |
|---|---|---|
| Basic plans | 56.99% | 13.99% |
| GILL | 54.18% | 16.19% |
| SGOLL | 55.63% | 14.84% |

Table 4.7: End-to-end navigation task completion rate for SGOLL.

is performing at least as well as the more time-consuming GILL algorithm.

Finally, we evaluate SGOLL on the end-to-end navigation task. Completion rates for both the single-sentence tasks and the complete tasks are shown in Table 4.7. While SGOLL does better than GILL on the single-sentence task, it is still worse than just training on the basic plans. Its performance on the complete task is worse than GILL although still slightly better than the basic plans. Overall this is not a particularly positive result but we will see in Section 4.6 later how to modify the algorithm slightly to improve its performance on this navigation task.

Having established that SGOLL is at least comparable to GILL on most of the tasks we evaluated on, we will next look at the computation times required by both algorithms. The timing results for the three different splits of data as well as the average times are shown in Table 4.8. All the results

|  | Grid-L | Grid-Jelly | L-Jelly | Average |
|---|---|---|---|---|
| GILL | 1,544.25 | 2,670.98 | 2,467.65 | 2,227.63 |
| SGOLL | 146.5 | 245.34 | 223.54 | 205.13 |

Table 4.8: The time (in seconds) it took for each algorithm to run on the different data splits. The last column shows the average time across the three data splits.

are obtained running the algorithms on Dell PowerEdge 1950 servers with 2x Xeon X5440 (quad-core) 2.83GHz processors and 32GB of RAM. Here SGOLL has a decidedly large advantage over GILL, requiring an order of magnitude less time to run.

### 4.4.2 Discussion

We have introduced an alternative lexicon learning algorithm that is much faster than GILL and potentially could scale to learning from much larger datasets. Moreover, its performance is generally comparable to that of GILL. As we will see in the next section, by making some further modifications to the lexicon learning algorithms, SGOLL even surpasses the performance of GILL in many instances.

One thing to note though is that while SGOLL makes the lexicon learning step much faster and scalable, another bottleneck in our overall system is training the semantic parser. Existing semantic parser learners such as Krisp or Wasp were not designed to scale to very large datasets and have trouble training on more than a few thousand examples. This remains an open problem for future research.

## 4.5 Modifying the Basic Framework

In this section we look at other, more minor ways of improving our systems that do not require changing our entire learning algorithms. We first look at adding the constraint of minimum support to the lexicon learning algorithms. In other words, we require an n-gram to be observed at least $m$ times before we will create a lexicon entry for it. We then look at learning higher-order n-grams beyond just unigrams and bigrams. Finally, we examine how changing the context-free grammar for our formal navigation plan language SAIL allows KRISP to learn better semantic parsers.

### 4.5.1 Requiring Minimum Support

One of the issues with the current lexicon learning algorithms is that often a lexical entry consisting of a rare n-gram and a rare graph will receive a very high score. Recall that the scoring function we use to rank the lexical entries is as follows for an n-gram $w$ and a graph $g$:

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

If an n-gram $w$ that only appears once in the training data co-occurs with a graph $g$ that also only appears once in the data, the pair $(w, g)$ will receive a perfect score of 1. Since we use a greedy approach of selecting the highest scoring lexical entries first, having such entries in the lexicon could potentially cause us to skip over another high-scoring lexical entry that has a lot more evidence supporting it. One way to alleviate the problem is to add smoothing

| Minimum | GILL | | | SGOLL | | |
|---|---|---|---|---|---|---|
| Support ($m$) | Precision | Recall | F1 | Precision | Recall | F1 |
| 0 | 78.54 | **78.10** | 78.32 | 82.91 | 71.28 | 76.65 |
| 10 | 91.52 | 68.80 | **78.49** | 86.48 | **72.18** | 78.68 |
| 20 | 94.04 | 64.76 | 76.61 | 87.30 | 72.09 | **78.97** |
| 30 | 94.86 | 60.49 | 73.74 | 87.90 | 70.80 | 78.43 |
| 50 | 96.30 | 52.56 | 67.86 | 88.19 | 67.58 | 76.52 |
| 100 | **97.86** | 41.99 | 58.62 | **89.10** | 60.33 | 71.94 |

Table 4.9: Accuracy of the inferred plans when requiring minimum support in the lexicon learning step. The highest value in each column is shown in bold.

| Minimum | GILL | | | SGOLL | | |
|---|---|---|---|---|---|---|
| Support ($m$) | Precision | Recall | F1 | Precision | Recall | F1 |
| 0 | 90.16 | **55.41** | **68.59** | 90.04 | 55.06 | 68.31 |
| 10 | 95.86 | 51.59 | 66.96 | 91.81 | **55.64** | **69.25** |
| 20 | 97.34 | 49.91 | 65.90 | 91.99 | 54.67 | 68.55 |
| 30 | 97.72 | 47.60 | 63.94 | 92.81 | 53.81 | 68.06 |
| 50 | 98.15 | 43.35 | 60.10 | 92.52 | 50.59 | 65.33 |
| 100 | **98.92** | 37.48 | 54.30 | **94.44** | 46.10 | 61.93 |

Table 4.10: Accuracy of the trained semantic parsers when requiring minimum support in the lexicon learning step.

to the estimated probabilities. Here we take an even simpler approach of requiring minimum support for an n-gram before we add it to our lexicon. This constraint can easily be added to both the GILL and SGOLL algorithms. For GILL, we do not try to compute the meaning of an n-gram if its initial set of candidate meanings has size less than $m$. For SGOLL, we skip over any n-gram that has counts less than $m$ when we are computing the lexicon. We test the performance of both algorithms on refining the landmarks plans.

| Minimum | GILL | | SGOLL | |
|---|---|---|---|---|
| Support ($m$) | Single-sentence | Complete | Single-sentence | Complete |
| 0 | 54.18% | **16.19**% | 55.63% | 14.84% |
| 10 | **56.25**% | 14.46% | **57.19**% | 17.90% |
| 20 | 55.94% | 13.46% | 56.85% | **18.04**% |
| 30 | 52.35% | 8.18% | 55.57% | 16.68% |
| 50 | 49.34% | 4.52% | 53.23% | 11.94% |
| 100 | 43.86% | 1.92% | 51.26% | 7.45% |

Table 4.11: End-to-end navigation task completion rate when requiring minimum support in the lexicon learning step.

The results on the three tasks: inferring navigation plans, training the semantic parser, and executing navigation plans, are shown in Tables 4.9, 4.10, and 4.11, respectively. Increasing the minimum support required generally increases the precision of the inferred plans but lowers the recall, a result that carries over to the semantic parsing results as well. This is as expected since we are basically throwing away entries from our original lexicon. Consequently, it is likely we will mark less of the nodes in the landmarks plans. One exception is when we add a minimum support of 10 to SGOLL, which actually resulted in slightly better recall and precision. For the navigation task, requiring minimum support improved the single-sentence performance for GILL but decreased the complete task performance. On the other hand, SGOLL performed better in both settings and even outperforms GILL now. One thing to note is that the two algorithms do not always count the number of occurrences for a n-gram $w$ the same way. GILL counts the size of the set of the initial candidate meanings, thus potentially counting multiple occurrences of

$w$ only once if they are all associated with the same graph. This is because we want to ensure that we have observed $w$ in many different contexts so we can generate a good list of intersections. However, the same is not necessary for SGOLL which does not rely on intersections to generate candidate meanings. Overall, requiring at least some minimal support seems to be beneficial. Thus, we set $m$ to be 10 for the rest of the experiments.

### 4.5.2 Varying the Maximum N-grams Learned

So far we have only looked at learning the meanings of unigrams and bigrams. Part of the reason we initially avoided trying to learn higher-order n-grams is due to the sparsity of the data as we increase n. As mentioned already, rare n-grams often resulted in high scoring entries in the lexicon due to the lack of negative evidence. However, since we added the requirement of minimal support, this is less of a concern. In general, trying to learn higher-order n-grams should at worst result in no changes in the lexicon as they would not have the requisite support. We again ran the same experiments as before, this time varying the maximum n-grams learned by GILL and SGOLL.

The results on the three tasks: inferring navigation plans, training the semantic parser, and executing navigation plans, are shown in Tables 4.12, 4.13, and 4.14, respectively. In general, increasing the maximum n-grams improved the results on all three tasks. At worst, it made no difference when we try to learn the higher-order n-grams. We set $n$ to be 4 for the rest of the experiments as it empirically gave the best results overall.

| Maximum | GILL | | | SGOLL | | |
|---|---|---|---|---|---|---|
| N-Gram | Precision | Recall | F1 | Precision | Recall | F1 |
| 1 | **93.27** | 65.84 | 77.12 | 85.24 | **74.96** | **79.76** |
| 2 | 91.52 | 68.80 | 78.49 | 86.58 | 72.38 | 78.85 |
| 3 | 91.45 | **70.44** | **79.51** | 87.22 | 73.20 | 79.60 |
| 4 | 91.48 | 70.37 | 79.48 | **87.32** | 72.96 | 79.49 |
| 5 | 91.48 | 70.36 | 79.47 | 87.27 | 72.61 | 79.27 |

Table 4.12: Accuracy of the inferred plans when varying the maximum n-grams learned in the lexicon learning step.

| Maximum | GILL | | | SGOLL | | |
|---|---|---|---|---|---|---|
| N-Gram | Precision | Recall | F1 | Precision | Recall | F1 |
| 1 | **97.74** | 50.01 | 66.05 | 90.21 | **56.70** | **69.61** |
| 2 | 95.86 | 51.59 | 66.96 | 91.50 | 55.24 | 68.85 |
| 3 | 95.78 | 53.02 | 68.15 | 91.86 | 55.55 | 69.22 |
| 4 | 95.80 | **53.48** | **68.54** | **92.22** | 55.70 | 69.43 |
| 5 | 95.77 | 52.91 | 68.07 | 91.96 | 55.26 | 69.02 |

Table 4.13: Accuracy of the trained semantic parsers when varying the maximum n-grams learned in the lexicon learning step.

| Maximum | GILL | | SGOLL | |
|---|---|---|---|---|
| N-Gram | Single-sentence | Complete | Single-sentence | Complete |
| 1 | 55.90% | 13.05% | 55.96% | 15.66% |
| 2 | 56.11% | 14.52% | 56.30% | 17.38% |
| 3 | 56.81% | **17.53%** | 56.67% | 16.65% |
| 4 | **57.22%** | 17.33% | **57.16%** | **17.56%** |
| 5 | 56.55% | 16.42% | 57.10% | 17.53% |

Table 4.14: End-to-end navigation task completion rate when varying the maximum n-grams learned in the lexicon learning step.

### 4.5.3 Changing the Context-Free Grammar for SAIL

Finally, we look at a change to the system that does not affect the lexicon learning process. Looking at all our results, we notice that even the semantic parsers trained on gold-standard data do not perform all that well. Thus, to boost the performance of our system, we would also need to improve the semantic parser learning step. Without altering KRISP or designing a new semantic parser learning algorithm, we look at changing the CFG we have used to parse the MRs. Since KRISP is learning when to apply a production rule in the CFG based on the natural language input, it is important that the production rules mirror the structure of the natural language (Kate, 2008). Thus, even using the same MRL (in this case, SAIL), changing its CFG could lead to improved semantic parsing performance.

Our original CFG for SAIL was designed to be compact with a small number of production rules. There were many recursive rules that can be used to generate an infinite number of actions or arguments. While these rules are quite expressive, they do not really correspond to any words or phrases in the NL. To alleviate this problem, we designed another CFG by expanding out many of the rules. While this resulted in many more production rules, each rule is better aligned with the NL. Details of both the original, compact CFG and the new, expanded CFG can be found in Appendix B.

Since this modification only affects the semantic parsing step, we only look at the results of two tasks: semantic parsing and navigation plan execution. We look at all the results we had training semantic parsers on different

| Navigation | Compact CFG | | | Expanded CFG | | |
|---|---|---|---|---|---|---|
| Plan Type | Precision | Recall | F1 | Precision | Recall | F1 |
| Basic | 86.68 | 48.62 | 62.21 | 85.39 | 50.95 | 63.76 |
| Landmarks | 50.40 | 31.10 | 38.39 | 52.20 | 50.48 | 51.11 |
| GILL | 95.80 | 53.48 | 68.54 | 94.13 | 54.29 | 68.79 |
| SGOLL | 92.22 | 55.70 | 69.43 | 88.36 | 57.03 | 69.31 |
| Gold-standard | 88.20 | 71.74 | 79.11 | 90.09 | 76.29 | 82.62 |

Table 4.15: Accuracy of the trained semantic parsers when using different CFGs for the MRL.

| Navigation | Compact CFG | | Expanded CFG | |
|---|---|---|---|---|
| Plan Type | Single-sentence | Complete | Single-sentence | Complete |
| Basic | 56.92% | 14.30% | 58.72% | 16.21% |
| Landmarks | 22.04% | 2.64% | 18.67% | 2.66% |
| GILL | 57.14% | 17.53% | 57.85% | 16.15% |
| SGOLL | 57.09% | 17.56% | 57.28% | 19.18% |
| Gold-standard | 58.24% | 26.51% | 62.67% | 29.59% |

Table 4.16: End-to-end navigation task completion rate when using different CFGs for the MRL.

plans, including the gold-standard annotations and see how they are affected when we switch to the expanded CFG. The results are shown in Tables 4.15 and 4.16. In general, using the expanded CFG produced better semantic parsers and also improved navigation results. Thus, in the rest of the experiments we use this expanded CFG instead of the original CFG.

## 4.6 Experimenting on Different Data

So far we have used the exact same training/testing data splits for all our experiments. In this section we will look at two other experiment scenarios. First, we will perform leave-one-instructor-out experiments. Instead of training on data from two maps and testing on the third, we will train on data from 5 out of the 6 instructors, and test on the 6th. This surprisingly resulted in much worse performance even though each training split contained more training examples. We will take a close look at the results to examine some of the reasons why this might have happened. The other experiment we will show is training and testing our system on Mandarin Chinese data. As was the case with the sportscasting task when we evaluated on Korean data, we want to demonstrate the generality of our system on the navigation task. The results for Chinese are about the same as for English, an encouraging sign that our system is indeed language-independent.

### 4.6.1 Cross-Instructor Experiments

The navigation corpus was collected from 6 different instructors over 3 different maps. The experimental results we have presented thus far have all split the data along the different maps. We train the system on examples from 2 of the 3 maps, then test the system on the held-out map. This gives us an idea of how the system might perform if we use it in an unseen, but relatively similar environment. However, another way to perform the cross-validation is to split the data among the different instructors. This would provide us with information about how the system might perform when it faces a new instructor.

To see how well our system could generalize across instructors, we ran the cross-instructor experiments, evaluating on the same three tasks as before. The results for inferring the navigation plans, training the semantic parser, and the end-to-end navigation task are shown in Tables 4.17, 4.18, and 4.19, respectively. On inferring the navigation plans, the performance is slightly better than in the cross-map experiments. This is partly due to the fact that each training split is larger in this scenario, composed of roughly $\frac{5}{6}$ of the total data rather than just $\frac{2}{3}$. However, the performances of the trained semantic parsers and consequently the overall navigation system are much worse than in the cross-map scenario.

We can gain some insight into why the cross-instructor scenario resulted in much worse performances by taking a closer look at the results. Table 4.20 shows the breakdown of the navigation task completion rate for each individual

| | Precision | Recall | F1 |
|---|---|---|---|
| GILL | 91.80 | 70.96 | 80.01 |
| SGOLL | 87.73 | 74.03 | 80.28 |

Table 4.17: Accuracy of the inferred navigation plans in the cross-instructor experiments.

| | Precision | Recall | F1 |
|---|---|---|---|
| GILL | 91.95 | 38.70 | 53.82 |
| SGOLL | 84.03 | 40.54 | 54.24 |

Table 4.18: Accuracy of the semantic parsers in the cross-instructor experiments.

| | Single-sentence | Complete |
|---|---|---|
| GILL | 35.33% | 3.02% |
| SGOLL | 37.71% | 5.25% |

Table 4.19: End-to-end navigation task completion rate for the cross-instructor experiments.

| Test | GILL | | SGOLL | |
|---|---|---|---|---|
| Instructor | Single-sentence | Complete | Single-sentence | Complete |
| WLH | 42.82% | 4.52% | 41.47% | 2.89% |
| EDA | 56.21% | 3.44% | 56.90% | 1.48% |
| EMWC | 53.37% | 2.34% | 68.98% | 14.76% |
| KLS | 16.44% | 1.31% | 16.14% | 3.52% |
| KXP | 27.29% | 3.20% | 24.19% | 5.60% |
| TJS | 15.85% | 3.31% | 18.58% | 3.22% |

Table 4.20: End-to-end navigation task completion rate broken down by each training-testing split in the cross-instructor experiments.

training-testing split. As can be seen, there is a huge variance between the performances on following navigation instructions from different instructors. For example, the single-sentence completion rates for the instructors WLH, EDA, and EMWC are much higher than for the other three instructors. This is evidence that there are a lot more differences between instructions provided by different people than between instructions provided by the same person for different, but similar environments. As we showed in some of the sample instructions in Section 4.1, there are many different styles of giving navigation instructions. Not only do they differ in the words or sentence structures used, they also differ in how they formulate the navigation plans. Given the small sample size (only trained on 5 instructor styles), our learning system has a hard time generalizing to unseen instructors. To make the system useful for a practical application, we would thus have to collect instructions from many different instructors rather than from just a few people.

### 4.6.2 Chinese Translation Data

One of the advantages of a learning system is that it can adapt to new scenarios given the proper training data. While our cross-instructor experiments showed that we did not have a sufficient number of different instructor styles in our data to properly generalize across them, here we look at adapting the system in another way. Since we take a language-independent approach, our system is able to learn a new language without any modifications to the system. Thus, we translated all of our English instructions into Mandarin

|  | Word Segmentation | Precision | Recall | F1 |
|---|---|---|---|---|
| GILL | By character | 90.90 | 73.21 | 81.02 |
| GILL | Stanford Segmenter | 91.79 | 71.41 | 80.23 |
| SGOLL | By character | 85.88 | 75.18 | 80.17 |
| SGOLL | Stanford Segmenter | 87.07 | 71.67 | 78.61 |

Table 4.21: Accuracy of the inferred plans for the Chinese corpus.

Chinese and tested to see if our system could indeed adapt to the translated training and testing data. The translation was done by a single native Chinese speaker.

One issue that does affect our system is the fact that Chinese is usually written without spaces between the words. Consequently, we would have to first segment the Chinese characters into words before we can apply our lexicon learning algorithms. We tried two different approaches. The first approach just naively inserts a space between every character, effectively treating each Chinese character as a word. While this may seem conceptually unpleasing at first, we are actually guaranteed to build lexical entries for all real Chinese words that contain less than 4 characters. This is because we are learning n-grams rather than just words. Thus, with the maximum n set to 4, we will consider all possible sequences of characters up to length 4. Of course, in this process we will also introduce many noisy lexical entries for sequences that start or end in the middle of words. The second approach we tried is using an existing tool to segment the characters. We used the Stanford Chinese Word Segmenter (Chang, Galley, & Manning, 2008) to perform the segmentation.

|  | Word Segmentation | Precision | Recall | F1 |
|---|---|---|---|---|
| GILL | By character | 92.48 | 56.47 | 70.01 |
| GILL | Stanford Segmenter | 94.36 | 55.72 | 70.00 |
| SGOLL | By character | 87.95 | 61.20 | 72.16 |
| SGOLL | Stanford Segmenter | 88.87 | 58.76 | 70.74 |

Table 4.22: Accuracy of the semantic parsers trained on the Chinese data

|  | Word Segmentation | Single-sentence | Complete |
|---|---|---|---|
| GILL | By character | 57.27% | 16.73% |
| GILL | Stanford Segmenter | 57.53% | 15.93% |
| SGOLL | By character | 58.54% | 16.11% |
| SGOLL | Stanford Segmenter | 58.70% | 20.13% |

Table 4.23: Navigation task completion rates for the Chinese corpus

The results for the experiments can be seen in Tables 4.21, 4.22, and 4.23. The performance on all three tasks are similar to those for the English data, indicating that our system can indeed readily adapt to learn different languages. The results for our two different word segmentation approaches are also about the same, except for the complete navigation task where SGOLL trained on words segmented by the Stanford Segmenter achieved the best completion rate. Nevertheless, this is an encouraging sign that even without an external tool to perform the word segmentation, a simple approach like segmenting by character can produce comparable results.

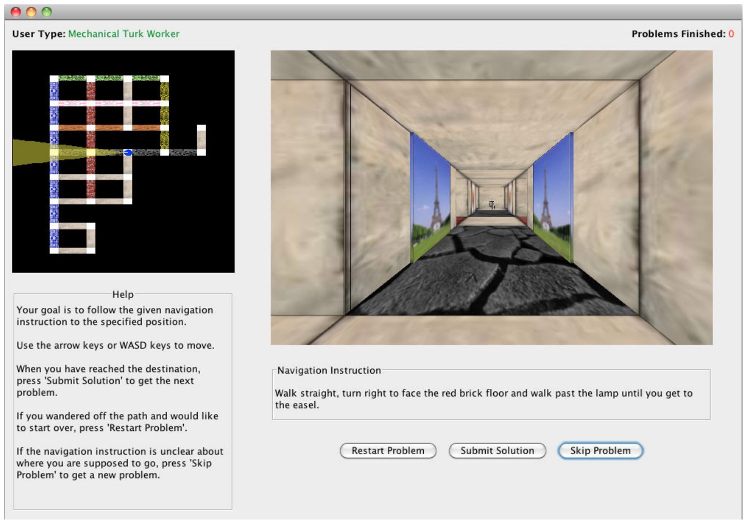
## 4.7 Collecting Additional Data using Mechanical Turk

One of the motivations for studying ambiguous supervision is the potential ease of acquiring large amounts of training data. Without requiring the annotations of MRs, a human only has to demonstrate how language is used in context. This is generally simple to do and consequently the system can be trained by anyone. We validate this claim by collecting additional training data for the navigation domain using Amazon's Mechanical Turk (Snow, O'Connor, Jurafsky, & Ng, 2008).

There are two types of data we are interested in collecting for the navigation task: natural language navigation instructions and follower data. Thus, we created two tasks on Mechanical Turk. The first one asks the workers to supply instructions for a randomly generated sequence of actions. The second one asks the workers to try to follow a given navigation instruction in our virtual environment. The latter task is used to generate the corresponding action sequences for instructions collected from the first task.

### 4.7.1 Task Descriptions

To facilitate the data collection, we first recreated the 3D environments used by MacMahon et al. (2006) for collecting the original data. This Java application allows the user to freely navigate our three worlds, Grid, L, and Jelly using the discrete controls of turning left, turning right, and moving forward one step. While we did not do so, we could also define new worlds with new objects and floor patterns with relative ease by supplying new 3D



Your goal is to follow the given navigation instruction to the specified position.

Use the arrow keys or WASD keys to move.

When you have reached the destination, press 'Submit Solution' to get the next problem.

If you wandered off the path and would like to start over, press 'Restart Problem'.

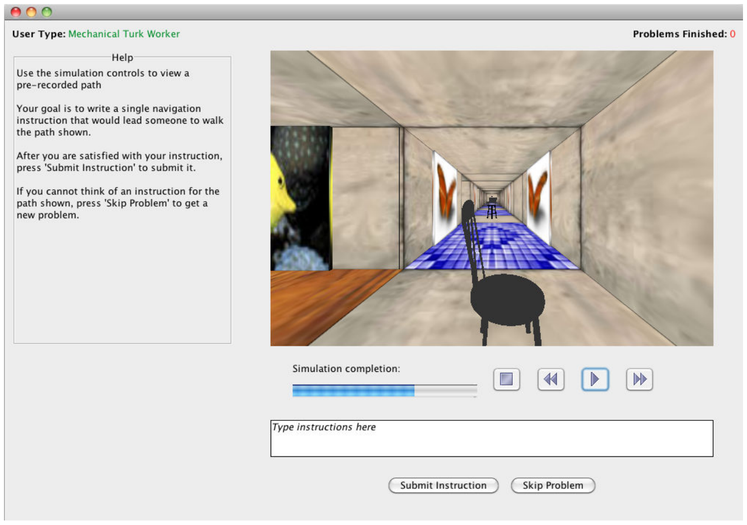
If the navigation instruction is unclear about where you are supposed to go, press 'Skip Problem' to get a new problem.

Walk straight, turn right to face the red brick floor and walk past the lamp until you get to the easel.

Figure 4.7: A screenshot of the follower task. Given a previously collected instruction, the worker is asked to follow that instruction to reach the intended destination.

mesh models or texture patterns.

The follower task is fairly straightforward using our application. The application first connects to our server requesting a follower problem to be solved which consists of an instruction and a starting location in a map. The worker is then placed at the starting location and asked to follow the navigation instruction as best as they could using the three discrete controls. When they have finished following the instruction, they press the submit button to get to the next problem. Alternatively, they could also skip the problem if they could not understand the instruction or if the instruction did not describe a viable route. A screenshot of this task can be seen in Figure 4.7. For each Human Intelligence Task (HIT), we asked the worker to complete 5 follower problems. We paid them $0.05 for each HIT, or 1 cent per follower problem.


User Type: Mechanical Turk Worker

Problems Finished: 0

Help

Use the simulation controls to view a pre-recorded path

Your goal is to write a single navigation instruction that would lead someone to walk the path shown.

After you are satisfied with your instruction, press 'Submit Instruction' to submit it.

If you cannot think of an instruction for the path shown, press 'Skip Problem' to get a new problem.

Figure 4.8: A screenshot of the instructor task. After watching a simulation of moving through a randomly generated path, the worker is asked to write an instruction that would lead someone to follow that path.

The instructions used for the follower problems were mainly collected from the Mechanical Turk instructor task. However, some of the instructions came from data collected by MacMahon (2007) that we previously did not use (hence did not segment into single sentences and align to the corresponding action sequences.)

The instructor task is slightly more involved because we ask the workers to provide new navigation instructions. The application again first connects to our server requesting an instructor problem. An instructor problem consists of a randomly generated action sequence in one of the worlds. The action sequences have a maximum length of 4 to keep the instructions short (we asked for a single sentence but this was not enforced). The worker is then asked to watch a simulation of the action sequence and provide instructions that would lead someone to perform those actions. This is similar to how Tellex et al. (2011) collected instruction data for their forklift application. A screenshot of this task is shown in Figure 4.8. Since this task requires more time to complete, each HIT consists of only 3 instructor problems. Moreover, we pay the workers \$0.10 for each HIT, or about 3 cents for each instruction they write.

One issue with collecting data using Mechanical Turk is quality control. This is especially problematic for the instructor task because there are no correct answers we can verify against. Consequently, we employ a tiered payment structure (Chen & Dolan, 2011) to reward and retain the good workers. The workers who have been identified to consistently provide good instructions were allowed to do higher-paying version of the same HITs that pay \$0.15 instead of \$0.10.

### 4.7.2 Data Statistics

Over a 2-month period we accepted 2,884 follower HITs and 810 instructor HITs from 653 workers. This corresponds to over 14,000 follower traces and 2,400 instructions. The total cost of the data collection was \$277.92. While there were 2,400 instructions, we needed to filter them to make sure they were of reasonable quality. First, we discarded any instructions that did not have at least 5 follower traces. Then we looked at all the follower traces and discarded any instruction that did not have sufficient follower agreement

|  |  |
|---|---|
| # instructions | 1011 |
| Vocabulary size | 590 |
| Avg. # words | 7.69 (7.12) |
| Avg. # actions | 1.84 (1.24) |

Table 4.24: Statistics about the navigation instruction corpus collected using Mechanical Turk. The average statistics for each instruction are shown with standard deviations in parentheses.

($> 0.5$). In other words, we require that the majority of the followers agree on a single path.

Using our strict filter, we were left with slightly over a thousand instructions. Statistics about this corpus can be seen in Table 4.24. Overall, this corpus has a slightly smaller vocabulary than the original data, and each instruction is slightly shorter both in terms of the number of words and the number of actions.

### 4.7.3 Using Mechanical Turk Data as Additional Training Data

One way to utilize the newly collected instructor and follower data is to use it to augment our existing training data. We added the corpus collected from Mechanical Turk to our original dataset and performed leave-one-map-out cross-validation. To make comparisons to earlier results easier, the test set only contains examples from the original dataset and not from the Mechanical Turk data.

The results of augmenting the original training data with the Mechanical Turk data is shown in Tables 4.25, 4.26, and 4.27. While there were not

| | Precision | Recall | F1 |
|---|---|---|---|
| GILL | 92.04 | 70.21 | 79.61 |
| SGOLL | 87.35 | 73.25 | 79.67 |

Table 4.25: Accuracy of the inferred plans for the original data using lexicon learned from the original data plus data collected from Mechanical Turk

| | Precision | Recall | F1 |
|---|---|---|---|
| GILL | 95.12 | 52.86 | 67.87 |
| SGOLL | 88.11 | 56.57 | 68.90 |

Table 4.26: Accuracy of the semantic parsers trained on the original data plus data collected from Mechanical Turk. The parsers are only tested on the original data.

| | Single-sentence | Complete |
|---|---|---|
| GILL | 58.16% | 18.30% |
| SGOLL | 57.62% | 20.64% |
| GILL (without Mechanical Turk data) | 57.85% | 16.15% |
| SGOLL (without Mechanical Turk data) | 57.28% | 19.18% |

Table 4.27: End-to-end navigation task completion rates for systems trained on the original data plus data collected from Mechanical Turk. The systems are only tested on the original data.

much difference in the first two tasks compared to training on just the original data, we see some improvements on the end-to-end navigation task. For both the single-sentence and the complete tasks, training with the addition data increased the performance. This is indication that the data we collected were indeed useful for solving the original problem even though the data were collected in very different manners from very different populations (controlled subjects versus Mechanical Turk workers).

### 4.7.4 Using Mechanical Turk Data as a Novel Test Set

Another way to utilize the Mechanical Turk data is to use it as a true test set since the data was never observed during training or development of our system. We train on all the original data and test on the Mechanical Turk data. This can be seen as another cross-instructor experiment as we are testing how well the system can generalize to unseen instructors.

The results for the experiments are shown in Tables 4.28 and 4.29. No semantic parsing results are reported since we do not have gold-standard annotations for the Mechanical Turk data. For the navigation task, there are no differentiations between the single-sentence and complete settings because all of the Mechanical Turk data were single sentences.[4] The completion rates are much worse than when we performed cross-map validation, although better than when we performed cross-instructor validation on the original data. This

[4]Some workers did write a few sentences, but we ignored the sentence boundaries and treated them as a single sentence

| | Precision | Recall | F1 |
|---|---|---|---|
| GILL | 92.27 | 71.69 | 80.69 |
| SGOLL | 87.99 | 74.33 | 80.59 |

Table 4.28: Accuracy of the inferred plans training on the entire original data collected by MacMahon et al. (2006)

| | |
|---|---|
| GILL | 39.59% |
| SGOLL | 40.53% |
| MARCO | 55.89% |

Table 4.29: Task completion rates testing on the newly collected instructions from Mechanical Turk

again confirms that we need to train on more instructor styles to properly generalize to unseen instructors. The relative positive performance compared to the previous cross-instructor experiment is likely due to the fact that the instructions in the Mechanical Turk corpus are shorter and simpler in general. The task completion rate of MARCO on this new test set is also provided as comparison. As expected, MARCO also performed significantly worse even though it does not do any learning. This is because MARCO was manually engineered using the original data and does not necessarily adapt to new instructors.

## 4.8 Discussion and Possible Extensions

Currently, our system goes through the various stages of learning in a pipelined manner. First, we learn a lexicon which is then used to create the supervised training data for the semantic parser learners. The semantic-parser learner then produces a semantic parser which is used during test time to transform new instructions into navigation plans. Finally, the executor carries out the navigation plans. As a result of this pipelined approach, 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 the MRs are executable actions, we can test our understanding of the language in the environment itself to receive additional reinforcements. For example, we could require that all refined plans must lead to the intended destinations when executed. However, it should be remembered that reaching the correct destination is not necessarily indicative of a good inferred plan (e.g. *landmarks plans* always lead to the correct destinations but do not correspond to the actual instructions.)

## 4.9 Chapter Summary

We have presented a novel framework for dealing with ambiguous supervision that is more sophisticated than the one in the previous chapter. By allowing any subgraph to be potentially aligned as the MR of a sentence, we are faced with an exponential problem where our previous approach of enumerating all the possible alignments and scoring them would not scale. In addition to the scalability problem, we also explicitly represent relationships between the different semantic entities with edges in a graph. This allows us to consider whether to align each semantic component to a NL sentence in conjunction with other components rather than make strong independence assumptions.

The key component of our framework is a lexicon learning algorithm called GILL that learns the meanings of n-grams by taking graph intersections of the potential meanings.

Applying our framework to the problem of learning to follow navigation instructions, we were able to infer most of the correct navigation plans associated with each instruction. Moreover, our system was able to reach the desired destination a majority of the time when the instructions are single sentences. We also introduced an alternative lexicon learning algorithm SGOLL that is an order of magnitude faster than GILL and has comparable performance. In addition to being faster, SGOLL also has the potential to scale to much larger datasets since it can be easily parallelized. Other modifications to our framework including requiring minimal support for an n-gram to be included in the lexicon and using a CFG for the MRL that is closer to natural language helped improve our system further.

In addition to testing our system using cross-map validation, we also tried cross-instructor validation. This resulted in much worse performance, which suggests that the variance between different instructors is higher than that between different, but similar environments. This is again confirmed when we tested on new data collected from Mechanical Turk. Thus, for building a practical application, we would need to collect instructions from many more different people for the system to generalize to unobserved instructors.

One advantage of our language-independent approach to language learning is that our system can adapt to learning new languages without any modifications. As we demonstrated in the previous chapter that our sportscasting system could learn to sportscast in Korean, we showed that our navigation system can learn to follow Mandarin Chinese navigation instructions. While Chinese introduces the problem of word segmentation, we showed that a naive approach of simply treating each character as a word performed about as well as first using an external word segmenter to determine the word boundaries.

Finally, we presented the results from collecting additional training data using Mechanical Turk. Since our system only requires training data in the form of language being used in a relevant context, virtually anyone can provide useful training data. In particular, for the navigation task, the teacher only has to demonstrate how to follow navigation instructions in a virtual environment without having to provide any semantic annotations of the instructions. Incorporating the additional data collected from Mechanical Turk into the training data resulted in the best end-to-end task completion rate on the original navigation problem.

# Chapter 5

# Related Work

In this chapter we will review relevant research work in the areas of robotics, computer vision, and computational linguistics. We will first look at the broad class of problems that aims to establish relationships between language and the world. The problem domains addressed span across a wide range including image captions, summaries of sports games, game playing and robot controllers, among many others. We will then review some recent work in dealing with the ambiguous supervision problem. Finally, we discuss some relevant work in semantic parsing and natural language generation.

Since language grounding encompasses a wide range of problems, we first categorize the existing work in this area by looking at the assumptions they make about the problems, including the type of supervision and the world context provided. We will then look at specific application areas these different approaches have addressed. As Deb Roy discusses in his theoretical framework for grounding language (Roy, 2005), the meaning of language can be divided into two types: referential and functional. Referential meanings talk about objects and events in the world in a descriptive manner. On the other hand, functional meanings aim to achieve some actions in the world. These are typically used in commands to cause the listener to perform particular actions. We divide the various application areas into these two large groups. Most of the earlier work focuses on learning referential meanings, or learning how to describe the world with words. However, some more recent work has explored using language as a guide to help accomplish certain tasks. We have looked at learning both types of meanings with our sportscasting and navigation applications. The sportscasting task requires the system to learn how to describe objects and events in the world. On the other hand, the navigation task requires the system to interpret instructions to perform certain actions in the world.

The main learning problem this thesis aims to solve is the ambiguous learning problem. So we will look at other approaches to tackling this problem including generative methods, ranking, and grammar induction. They all implicitly or explicitly solve the matching problem of determining the relevant parts of the world context that are referred to by the language.

Finally, we discuss work in the areas of semantic parsing and natural language generation. Since both of these areas are quite large, we will only look at the work most closely related to this thesis. In particular, we focus our attention on semantic parser learners that require less than full supervision and learning methods for performing surface realization.

The rest of the chapter is organized as follows. We first categorize existing work in grounded language learning along several common characteristics in Section 5.1. We then look at specific application areas that aim to learn referential and functional meanings in Sections 5.2 and 5.3, respectively. Next we review relevant work in solving the ambiguous supervision problem in Section 5.4. Finally, we discuss related work in semantic parsing and natural language generation in Sections 5.5 and 5.6, respectively.

## 5.1 Connecting Language and the World

While there are many different approaches to connecting natural language to the world, we can categorize them by looking at various assumptions they make about the problems. We discuss several different dimensions that we can characterize these approaches by below.

- **Type of supervision** Most work assumes some form of parallel data where the natural language is paired with a relevant grounding context to which the language refers to. However, they differ in how tightly coupled the context is to the language. The earliest work aims to learn names of objects/people and simple attributes (Siskind, 1996; Satoh, Nakamura, & Kanade, 1997; Roy, 2002; Barnard et al., 2003; Berg, Berg, Edwards, & Forsyth, 2004). Here the context is assumed to be static and contain direct representations of the words to be learned. Beyond these static scenarios, other work uses more dynamic context such as videos (Fleischman & Roy, 2007; Gupta & Mooney, 2009; Buehler, Everingham, & Zisserman, 2009) where temporal cues are used to get the approximate context in which an event occurred. Another dimension of complexity comes from grounding complete sentences rather than just words, which requires understanding words that are not directly represented in the context (Gorniak & Roy, 2005; Farhadi, Hejrati, Sadeghi, Young, Rashtchian, Hockenmaier, & Forsyth, 2010; Matuszek et al., 2010; Qu & Chai, 2010). Our work assumes both dynamic contexts and tries to ground complete sentences.

There is also work that does not use parallel data at all, instead using responses from the world to guide its learning (Branavan et al., 2009; Branavan, Zettlemoyer, & Barzilay, 2010; Vogel & Jurafsky, 2010; Branavan et al., 2011). These methods use reinforcement learning to establish the correspondence between language and the actions that lead to rewards (e.g. completing the desired task, winning a game, etc).

- **Representation of the grounding context** Depending on the end goal of the grounding task, different representations of the world are used which varies in their granularity and structure. For dealing with real world perceptions, often the raw pixels of an image or video frames are used as the representation of the world (Roy, 2002; Berg et al., 2004; Fleischman & Roy, 2007; Gupta & Mooney, 2009; Buehler et al., 2009). These raw visual features can then be additionally processed to identify certain structures. Examples include segmenting an image into different regions (Barnard et al., 2003) and detecting faces (Satoh et al., 1997; Berg et al., 2004).

  Our work does not use real world perceptions and assumes a symbolic representation of the world instead. Symbolic representations can range from simple state-action transitions (Branavan et al., 2009; Vogel & Jurafsky, 2010; Branavan et al., 2011), to records of events (Snyder & Barzilay, 2007; Liang et al., 2009), and to more hierarchical representations (Kollar et al., 2010; Tellex et al., 2011). The sportscasting task assumes a record structure where each record is independent of each other. On the other hand, the navigation task deals with more complicated relational data that models how entities in the world relate to each other.

- **Ambiguity level** For parallel data, one way to characterize the complexity of the alignment problem is by how ambiguous the training data is. The data could be unambiguous if alignment is provided as part of the training data (Kollar et al., 2010; Tellex et al., 2011). The data could exhibit small amounts of ambiguity as in the sportscasting task where a handful of possible representations can be aligned to (Siskind, 1996; Satoh et al., 1997; Roy, 2002; Barnard et al., 2003; Berg et al., 2004; Matuszek et al., 2010). Finally, if we allow for partial alignments as for the navigation task, there can be an exponential number of possible choices for each alignment (Liang et al., 2009).

- **End task** Since grounding is a complicated problem, various work has concentrated on different components of the problem. Some work has focused on solving the alignment problem (Siskind, 1996; Satoh et al., 1997; Roy, 2002; Barnard et al., 2003; Berg et al., 2004; Liang et al., 2009; Kim & Mooney, 2010; Bordes et al., 2010) where the goal is to find the correct correspondences between the language and the corresponding parts of the context. Moving beyond alignment, some work has evaluated on semantically parsing (Clarke et al., 2010; Liang et al., 2011; Borschinger, Jones, & Johnson, 2011) or generating (Kulkarni, Premraj, Dhar, Li, Choi, Berg, & Berg, 2011; Li, Kulkarni, Berg, Berg, & Choi, 2011) entire sentences. Both of our domains evaluate on semantic parsing and the sportscasting domain also tests on generating sentences.

For some tasks, the end goal is to perform certain tasks such as for our navigation domain (Branavan et al., 2009, 2010; Vogel & Jurafsky, 2010; Matuszek et al., 2010; Kollar et al., 2010; Tellex et al., 2011; Branavan et al., 2011). Here the interpretation of the language is used as guidance to accomplishing the overall task. The tasks can range from other navigation tasks to performing computer tasks to playing games. We will discuss more of the specific application areas in Section 5.3.

The differences in the nature of the problems addressed contribute to the different approaches developed. As the context becomes more complicated and ambiguous, often additional cues beyond just the language are required to learn well. Similarly, dealing with more sophisticated language often requires additional linguistic knowledge such as a syntactic parser or a partial lexicon. Our work tries to push the limit on what can be learned without using either form of additional prior information.

## 5.2 Learning Referential Meanings

Referential meanings are useful both for describing the world and for understanding descriptions of the world. The former allows a computer system to automatically generate linguistic reports about its knowledge and perceptions. The latter helps a computer system understand a visual scene from the descriptions. We will review work that deals with referential meanings as well as several problem domains that have generated a lot of interests, including captions of images, sports videos, and spatial relations.

One of the most ambitious end-to-end visually-grounded scene-description system is VITRA (Herzog & Wazinski, 1994) which comments on traffic scenes and soccer matches. The system first transforms raw visual data into geometrical representations. Next, a set of rules extract spatial relations and interesting motion events from those representations. Presumed intentions, plans, and plan interactions between the agents are also extracted based on domain-specific knowledge. However, since their system is hand-coded it cannot be adapted easily to new domains.

Srihari and Burhans (1994) used captions accompanying photos to help identify people and objects. They introduced the idea of visual semantics, a theory of extracting visual information and constraints from accompanying text. For example, by using caption information, the system can determine the spatial relationship between the entities mentioned, the likely size and shape of the object of interest, and whether the entity is natural or artificial. However, their system is also based on hand-coded knowledge.

Siskind (1996) performed some of the earliest work on *learning* grounded word meanings. His learning algorithm addresses the problem of ambiguous training or "referential uncertainty" for semantic lexical acquisition, but does not address the larger problems of learning complete semantic parsers and language generators.

Around early 2000s, several robotics and computer vision researchers started working on inferring grounded meanings of individual words or short referring expressions from visual perceptual context (e.g., Bailey et al., 1997; Roy, 2002; Barnard et al., 2003; Yu & Ballard, 2004). However, the complexity of the natural language used in this earlier work is usually very restrictive, with many of the systems using pre-coded knowledge of the language, and almost all use static images to learn language describing objects and their relations, and cannot learn language describing actions. The most sophisticated grammatical formalism used to learn syntax in this work is a finite-state hidden-Markov model. By contrast, our work exploits the latest techniques in statistical context-free grammars and syntax-based statistical machine translation that handle more of the complexities of natural language.

Some more recent work include the work by Gold and Scassellati (2007) and Buehler et al. (2009). Gold and Scassellati (2007) built a system called TWIG that uses existing language knowledge to help it learn the meaning of new words. The robot uses partial parses to focus its attention on possible meanings of new words. By playing a game of catch, the robot was able to learn the meanings of "you" and "me" as well as "am" and "are" as identity relations. Buehler et al. (2009) built a system that learned sign language from TV broadcasts. While hand signs are generally aligned with the subtitles temporally, the alignments are not exact. Thus, they use a sliding window to generate multiple sequences of hand signs that could be aligned to each word. This problem is similar to our sportscasting task in that they also use the weak temporal signals to construct candidates. However, they are only learning at the word level rather than at the sentence level. They solve the ambiguity problem using multiple instance learning by treating each word with all the possible candidate hand signs as positive bags. Negative bags can be constructed easily from video windows that are far from the target word.

The area of language acquisition is also of great interest to the psychology community. For example, Regier and Carlson (2001) presented a computational model for capturing geometric features of spatial relations. Frank, Goodman, and Tenenbaum (2009) studied the CHILDES corpus (MacWhinney, 2000) which included videos of two infants and their mothers playing with a set of toys. They built a model that simultaneously learns the speaker's intentions and a lexicon and showed improvement over other cross-situational learning methods. Fazly, Alishahi, and Stevenson (2010) also studied the same corpus and proposed a probabilistic incremental model for cross-situational word learning. Their algorithm can learn online similar to our algorithm SGOLL and was shown to exhibit many of the same word learning behaviors as those observed in children. They later extended their model to incorporate syntactic knowledge as well to further improve the performance of the model

(Alishahi & Fazly, 2010). However, in their experiments, the semantic representations were artificially generated from the text itself rather than from the environment and do not encode any relational data. Moreover, the referential ambiguities were also artificially introduced by including the correct semantic representations of adjacent sentences.

### 5.2.1 Captions of Images and Videos

One application that has received a lot of attention, especially from the computer vision community, is the task of learning from captions that accompany photos or videos (Satoh et al., 1997; Berg et al., 2004). This area is of particular interest given the large amount of captioned images and videos available on the web including from news articles and photo-sharing sites such as Flickr.

Early work mainly used the captions to identify names of people in the images. Satoh et al. (1997) built a system to detect faces in newscasts and associate them with names in the captions. They used fairly simple manually-written rules to determine how to assign the names to the faces. Berg et al. (2004) used a more elaborate learning method to cluster faces in news articles and label the clusters with names. Using contextual features, they estimated the likelihood a person named in the captions actually appears in the given image. This helps improve the precision of their system by avoiding trying to attach all the names in a caption to a face.

Beyond just names of people, captions also provide cues for what objects are depicted in the images. Morsillo, Pal, and Nelson (2009) devised a semi-supervised learning approach that uses both visual and textual features of web documents to determine whether an image contains an object of interest. Their system first retrieves a set of candidate images using an image search engine and then refines the set of returned results based on the image itself and nearby text. They then use the set of filtered images as training data to build visual classifiers.

Recently, there has also been some interest in the task of generating captions. Given an image, the goal is to produce a sentence that describes the content of the image. Farhadi et al. (2010) constructed a dataset of image descriptions by asking Mechanical Turk workers to write sentences describing selected images from the 2008 PASCAL development kit. They also annotated each image with a triplet of <object, action, scene> which serves as the common meaning space between the images and the NL descriptions. Similarity between an image and a description can be measured by first mapping both to these triplets. This allows retrieval of an image based on a description and vice versa. Similar work was done by Feng and Lapata (2010) who defined a single distribution model for both visual and textual features. They learn to generate captions for images in news articles by selecting sentences or phrases from the accompanying text. While these approaches rely on existing text that describes the images, there has also been work on generating novel sentences using various scoring metrics (Kulkarni et al., 2011; Li et al., 2011). They first use visual detectors to determine the subject, object, and action depicted in the images. They then fuse these components together using templates, language models, or web-scale n-grams to produce fluid and coherent descriptions.

In addition to captions providing cues for what are in the images, images can also help build the semantics of language. Bruni, Tran, and Baroni (2011) combined text and image features into a single lexical semantic model. These combined features can then be used for similarity judgements between two words.

### 5.2.2 Sporting Events

Sports videos is another interesting domain for grounded language acquisition due to the abundance of data. Sportscasts and articles about sports games naturally contain lots of descriptions about the events that occurred. Moreover, the events are somewhat structured according to the rules of the sport. Some recent work on video retrieval has focused on learning to recognize events in sports videos (baseball and ice skating) and connecting them to English words that appear in the accompanying closed captions (Fleischman & Roy, 2007; Gupta & Mooney, 2009). However, these projects only learn the connections between individual words and video events and do not learn to describe events using full grammatical sentences. Our sportscasting task, on the other hand, avoids the computer vision issues in processing the videos, and concentrates on the language issues instead.

There has also been some work on aligning sentences in sports articles to box scores from the game (Snyder & Barzilay, 2007; Liang et al., 2009). Snyder and Barzilay (2007) used a supervised approach to learn the correspondence between text in recaps of American football games and database records that contain statistical information about the game (e.g. passing yards, number of touchdowns, etc). Liang et al. (2009) also built a system that performed alignment on this data, but used a generative model that did not require annotations of the correct correspondences.

### 5.2.3 Spatial Relations

When describing objects in the world, the spatial relations between them are of particular importance for constructing referring expressions. In addition to identifying and describing different attributes of an object, it is often helpful or necessary to specify its spatial relationships to other objects in order to disambiguate among similar objects. Regier and Carlson (2001) studied how humans perceive these spatial relations and built a computational model called the attention vector-sum (AVS) model that captures the semantics of various spatial and geometric relations. This provides a quantitative way of defining these spatial relations.

Beyond just understanding how to describe specific spatial relations, the pragmatics of selecting the most useful set of spatial relations to describe are also important. Golland, Liang, and Klein (2010) devised a language game where the goal is for the speaker to identify a particular object in a scene, and for the listener to select that object. The speaker model that contains an embedded listener model produced the best result because it took into account how well the spatial descriptions can help the listener disambiguate among the set of objects.

Spatial relations are also useful when describing certain actions. For example, Tellex and Roy (2009) built a system that learns to identify human locomotive actions in videos that could be used for video retrieval. They studied prepositional phrases such as to, across, through, out, along, towards, and around to help determine how the person in the video is moving in relation to the other objects. They collected training data by asking the annotators to describe the human motion in the videos by completing the sentence "the person is going ... ".

## 5.3 Learning Functional Meanings

Instead of just describing the world, functional meanings are used to induce actions that affect the world. Recently, there has been a rise in the number of projects that use language to aid computer agents in completing certain tasks. The type of tasks include playing games, controlling mobile robots, and performing other computer-related tasks such as setting up an email account. In most cases, the language is used to instruct the computer agents on what to do, either explicitly (e.g. "press the OK button", "go down this hallway", etc) or as high-level advice (e.g. "build cities near rivers", "go back to the charging station when the battery is low", etc). In more interactive settings, the computer agent must also be able to respond in natural language (e.g. "the soup of the day is a french onion soup") or prompt the user to perform certain tasks (e.g. "I am holding the lever that releases the door so you should be able to get out now") or to ask clarification questions (e.g. "should I press the red button on the left or on the right"). Our navigation task falls into this category of understanding functional meanings as the goal is to learn to follow navigation instructions to reach the desired destination. We will review work in several application domains that deals with functional meanings in the following sections.

### 5.3.1 Controlling Mobile Robots

Building computer systems that can interpret navigation instructions is particularly important to robotics, where such a system would provide a natural interface to controlling mobile robots. For example, Kruijff, Zender, Jensfelt, and Christensen (2007) designed a robot that uses human linguistic inputs to help it build a topological map. The robot also asks the human to clarify uncertainties such as whether there is a door in the nearby vicinity. The system uses a hand-built combinatory categorial grammar to parse the language.

Our work on the navigation task is the most similar to that of Matuszek et al.'s (2010). Their system learns to follow navigation instructions from example pairs of instructions and map traces with no prior linguistic knowledge. They used the semantic-parser learner WASP (Wong & 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 the 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, Bader, Bard, Boyle, Doherty, Garrod, Isard, Kowtko, McAllister, Miller, Sotillo, Thompson, & Weinert, 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. Finally, Tellex et al. (2011) built a forklift simulator that could interpret natural language commands such as picking up pallets and loading them onto specific trucks. They crowdsourced the training data by showing the annotator a short simulation and asking them to write down commands that would lead to the observed actions. They then manually annotated the spatial description clauses contained in these commands.

Since the focus of most of this work is on building controllers for robotic systems, they are not as concerned with the language learning issues and usually rely on some amount of external linguistic information such as object names, predefined spatial terms (e.g. above, right, through, etc), or manual annotations. In contrast, our approach does not assume any existing linguistic knowledge or resources and can be readily applied to languages other than English.

### 5.3.2 Playing Games

In addition to navigation tasks, there has also been recent work on learning to perform other tasks such as puzzle solving or playing computer games. The computer can play several different roles in these games. It can take instructions or advice from humans to perform certain tasks in the games, it can provide guidance to the human player by giving hints or providing instructions, and it can also participate as another player in the game and interact or cooperate with real human players.

As with the case of building navigation systems, most of the early work in this domain focus on solving system-building issues and usually rely on language-specific resources such as syntactic parsers or predefined lists of objects and actions. Gorniak and Roy (2005) built a computer player that could communicate and cooperate with human players in order to jointly accomplish a task in a role-playing game. The players have to navigate to certain locations and perform various actions such as pushing levers and lighting torches in the correct sequence in order to complete the task. The system first syntactically parses the spoken instructions and then maps them to specific actions. Kerr, Cohen, and Chang (2008) developed a system that learns grounded word-meanings for nouns, adjectives, and spatial prepositions while a human is instructing it to perform tasks in a virtual world. This system also assumes an existing syntactic parser. It also relies on prior knowledge of verb semantics and is unable to learn these from experience.

In contrast to these goal-orientated games, Orkin and Roy (2009) developed a more open-ended game called the Restaurant Game where there are no clearly defined objectives to achieve. One player assumes the role of a customer in a restaurant while another player assumes the role of the waitress. While the players generally follow a typical restaurant script, they can also deviate from the normal behavior and act as they please (e.g. fill the tables with wine glasses). The goal of the project is to produce a computer player that learns to act and talk like human players. Some work has been done to learn to associate words to objects in this game (Reckman, Orkin, & Roy, 2010).

When humans learn language, they are typically able to use additional cues beyond just the language and the perceptual environment. Qu and Chai (2010) builds upon the work by Yu and Ballard (2004) and incorporates user language behavior, domain knowledge, conversation context, and eye gaze to help it learn language. Set in a treasure hunting game, the agent learns to carry on conversations with the human player.

Most recently, Branavan et al. (2011) presented a system that learns to play a complex turn-based strategy game by reading the instruction manual for the game. The game requires sophisticated strategies and long-term planning that are difficult even for humans to master. They showed that by incorporating the knowledge gleaned from the manual, the system improved its gameplay significantly .

### 5.3.3 Performing Computer Tasks

Other than navigation and game playing, there has also been some interest in learning how to interpret English instructions for performing computer tasks. Lau et al. (2009) collected a corpus of instructions using Mechanical Turk for performing common tasks on the web such as performing a search or signing up for an email account. Their goal is to build a system that can assist users in following natural-language instructions. The actions in this domain include going to certain URLs, clicking on links, and entering information into a web form. They compared three different approaches, with the keyword-based approach outperforming the grammar-based and the machine learning approaches on identifying the correct action to perform.

Working on a similar task, Branavan et al. (2009) built a system that could follow troubleshooting instructions for a Windows machine. They use a reinforcement learning approach to map instructions into low-level GUI actions such as clicking on buttons and menus or typing into text fields. They later expanded this framework to also learn high-level instructions that do not directly correspond to a single GUI action (e.g. “open control panel” requires a series of clicks on different menu items) (Branavan et al., 2010). They learn a model of the environment guided by their instruction interpreter. The environment model in turn helps them compute look-ahead features that assess the longer-term consequences of an action. Overall, the system was able to correctly interpret high-level instructions a majority of the time and its performance on the low-level instructions also improved compared to their previous model.

## 5.4 Learning from Ambiguous Supervision

One of the core problems this thesis addresses is how to learn from ambiguous supervision where it is not clear what the correct annotations are. This can be formulated as an alignment problem where the goal is to match sentences to facts in the world to which they refer. As mentioned previously, there has been work aligning text from English summaries of American football games to database records that describe events and statistics about the game (Snyder & Barzilay, 2007; Liang et al., 2009). Snyder and Barzilay (2007) use a supervised approach that requires annotating the correct correspondences between the text and the semantic representations. On the other hand, Liang et al. (2009) have developed an unsupervised approach using a generative model to solve the alignment problem. They also demonstrated improved results on matching sentences and events on our RoboCup English sportscasting data. However, their work does not address semantic parsing or language generation. Section 3.6 presented results showing how we could initialize our system with the output of their algorithm to achieve better results on matching, semantic parsing, and language generation. Kim and Mooney (2010) further improved the generative alignment model from Liang et al. (2009) by incorporating the full semantic parsing model from Lu et al. (2008). This results in a joint generative model that outperforms all previous results on the matching and language generation tasks and produces competitive performance on the semantic parsing task in our sportscasting domain.

In addition to treating the ambiguous supervision problem as an alignment problem, there have been other approaches such as treating it as a ranking problem (Bordes et al., 2010), or a PCFG learning problem (Borschinger et al., 2011). Similar to how we solved the superfluous comment problem in Section 3.7, Bordes et al. (2010) use the idea that semantic tokens in the candidate sets should generally be preferred to ones that are not. Thus, they use a supervised ranking approach that optimizes to rank the semantic tokens in the candidate sets above tokens selected randomly. The learned ranker is then used to select the best token in the candidate set. They showed improved results on the matching task for our sportscasting data as well as another dataset about weather forecasts. Borschinger et al. (2011) takes a very different approach that reduces the grounded language learning task to a grammatical inference problem. They created a PCFG that incorporates information about both the forms and the meanings of the sentences. The PCFG contains context symbols for each observed set of candidate meanings and rules that map every semantic concept to every word in the data. The context symbols restrict which MRs might parse into a particular string but are marginalized out during testing since the set of candidate meaning are not available. The weights of the PCFG is learned using the Inside-Outside algorithm. They showed competitive results on the semantic parsing task on the sportscasting data and surpassed the previous result when they augmented their model to learn the canonical word order in the language as well.

## 5.5 Semantic Parsing

As mentioned in Chapter 2, most existing work on semantic-parser learning has focused on supervised learning where each sentence is annotated with its semantic meaning. Some semantic-parser learners additionally require either syntactic annotations (Ge & Mooney, 2005) or prior syntactic knowledge of the target language (Ge & Mooney, 2009; Zettlemoyer & Collins, 2005, 2007). Since the world never provides any direct feedback on syntactic structure, language-learning methods that require syntactic annotation are not directly applicable to grounded language learning. Therefore, methods that learn from *only* semantic annotation are critical to learning language from perceptual context.

While we use logic formulas as our MRs, the particular MRLs we have used contain only atomic formulas and can be equivalently represented as frames and slots. There are systems that use transformation-based learning (Jurcicek, Gasic, Keizer, Mairesse, Thomson, & Young, 2009), or Markov logic

(Meza-Ruiz, Riedel, & Lemon, 2008) to learn semantic parsers using frames and slots.

Both of the semantic parsers we have used, KRISP and WASP, require a CFG for the MRL. As seen in Section 4.5.3, the particular choice of CFG can greatly affect the quality of the semantic parsers learned depending on how close the CFG mimic the structure of natural language. Lu et al. (2008) introduced a more flexible generative model that does not make use of an explicit grammar. Instead, it models the correspondence between NL sentences and MRs with a generative model using hybrid trees. The implicit grammar representation leads to models that generalize well.

One of the reason for the low performance of our navigation system on the complete task is that we parse each sentence in the instruction independently. Zettlemoyer and Collins (2009) showed that by taking into account the context in which a sentence appears, they can achieve better semantic parsing results.

Parallel to our work on building semantic parsers from ambiguous supervision, other recent work has also looked at training semantic parsers from supervision other than logical-form annotations. Clarke et al. (2010) and Liang et al. (2011) describe approaches for learning from question and answer pairs. They automatically find semantic interpretations of the questions that would generate the correct answers. This is similar to our navigation task where we have to infer the semantic interpretations of the navigation plans that would lead to the correct action sequences. Artzi and Zettlemoyer (2011) use conversation logs between a computer system and a human user to learn to interpret the human utterances. While the meaning of the human-side of the conversation is unknown, the state of the system and the semantic interpretation of its utterances are all available. In a conversation, the system can ask clarification questions which helps make clear the meaning of the prior human utterances.

There have also been some unsupervised approaches to semantic parsing. Poon and Domingos (2009, 2010) introduced an unsupervised system for extracting knowledge from raw text. They automatically cluster semantically equivalent dependency tree fragments, and identify their predicate-argument structures. While their system is useful for applications such as question answering, they do not parse the language into a pre-defined formal language. On the other hand, Goldwasser et al. (2011) presented an unsupervised approach of learning a semantic parser by using an EM-like retraining loop. They use confidence estimation as a proxy for the model's prediction quality, preferring models that have high confidence about their parses.

## 5.6 Natural Language Generation

The research area of NLG address many different problems related to language generation, ranging from long-term conversation planning to more local problems such as lexical choice and word order. We have mostly focused on the surface realization issues of learning how to map from concepts to natural language. To generate complete sportscasts, we have also looked at the content selection problem of deciding the relevant facts to talk about.

Evaluating generation can be difficult as there are usually many acceptable solutions. Moreover, subtle differences in the generation quality can be difficult to quantify. The recently introduced Generating Instructions in Virtual Environments (GIVE) challenge aims to solve this problem by providing an end-to-end generation task (Byron, Koller, Striegnitz, Cassell, Dale, Moore, & Oberlander, 2009).[1] Each generation system is responsible for providing instructions to human players to help them solve a treasure hunting puzzle game. The quality of the generation systems can be quantified by how fast the players complete the puzzles. Our navigation task was partly inspired by this challenge in that we wanted to construct a task-based evaluation that could quantify the performance of an entire system.

There are several existing systems that generate sportscasts for RoboCup games (André et al., 2000). Given game states provided by the RoboCup simulator, they extract game events and generate real-time commentaries. These systems take into account many practical issues such as timeliness, coherence, variability, and emotion that are needed to produce good sportscasts. However, these systems are all hand-built and generate language using predetermined templates and rules. In contrast, we concentrate on the learning problem and induce the generation components from ambiguous training data. Nevertheless, augmenting our system with some of the other components in these systems could improve the final sportscasts produced.

[1]We participated in the first GIVE challenge by submitting a simple template system.

As mentioned in Section 2.2, we use WASP$^{-1}$ to learn a natural language generator for our sportscasting task. Using the same SCFG rule we use for semantic parsing, WASP$^{-1}$ can also translate a MR into a NL sentence. There is also prior work on learning a lexicon of elementary semantic expressions and their corresponding natural language realizations (Barzilay & Lee, 2002). This work uses multiple-sequence alignment on datasets that supply several verbalizations of the corresponding semantics to extract a dictionary.

Similar to how WASP defines a single model that can support both semantic parsing and natural language generation, the hybrid tree generative model proposed by Lu et al. (2008) can also be used for both tasks. By using tree conditional random fields to parameterize the model to capture some longer range dependencies, the inverted hybrid tree model was shown to outperform WASP$^{-1}$ (Lu et al., 2009).

Prior to transforming a MR into a NL utterance, the system must first decide what to say (content selection). Duboue and McKeown (2003) were the first to propose an algorithm for learning content selection automatically from data. Using semantics and associated texts, their system learns a classifier that determines whether a particular piece of information should be included for presentation or not.

There has been some work on learning content selection using reinforcement learning (Zaragoza & Li, 2005). They use a setting similar to our navigation task in a video game environment where the speaker must aid the listener in reaching the destination while avoiding obstacles. They repeatedly played the game to find an optimal strategy that conveyed the most pertinent information while minimizing the amount of messages. We consider a different problem setting where such reinforcements are not available to our content selection learner.

In addition, there has also been work on performing content selection as a collective task (Barzilay & Lapata, 2005). By considering all the content selection decisions jointly, they capture dependencies between each uttered items. This creates more consistent overall output and aligns better with how humans perform on this task. Such an approach could potentially help our system produce better overall sportscasts.

While content selection and surface realization are typically handled separately, Angeli et al. (2010) introduced a unified framework for performing both tasks. They break up the end-to-end generation process into a sequence of local decisions and train each discriminatively. They achieve competitive results in three domains, including our RoboCup sportscasting domain.

# Chapter 6

# Future Work

The work presented in this thesis has only started to scratch the surface of the problem of grounded language learning. While we have made some progress toward solving the ambiguous supervision problem, there is still a lot of room for improvements both in terms of the overall performance of the learning algorithms and the scalability of the systems to larger datasets. In this chapter we will describe both immediate potential extensions to our work, and also some longer-termed research ideas in this area.

## 6.1 Algorithmic Improvements to Learning from Ambiguous Supervision

Since the central question we are concerned with in this thesis is how to resolve the ambiguous supervision problem to learn the semantics of language, we will first look at some potential algorithmic improvements to our learning systems.

### 6.1.1 Integrating Different Components

Both our sportscasting and navigation systems are ad-hoc systems that do not have unified frameworks. We have mainly focused on resolving the ambiguity in the training data to create traditional, supervised semantic annotations. The other components of the systems were created to build a working system and are not integrated into the learning algorithms. For example, the content selection and surface realization steps are dealt with separately and each component is trained without regards to the other. However, conceptually they are much more closely related, and solving one problem should help solve the other. As shown by Liang et al. (2009), using a single generative process that selects the MRs to express and produces the actual words resulted in better performance on the matching task. Kim and Mooney (2010) further showed that by additionally incorporating a semantic parser into the model, the system performed better on the other tasks as well.

Similarly for the navigation system, the lexicon-learning step is not integrated with building the semantic parser. This is actually quite odd as the semantic-parser learner also has to build a lexicon, either explicitly or implicitly. A more natural way to build the system would be to utilize the lexicon we learned already and to only learn compositional rules to create a semantic parser. The executor could also provide feedback to the learning algorithms. As mentioned in Section 4.8, refined plans that do not lead to the intended destinations are generally indicative of errors in the refinement step.

Part of the reason for the ad-hoc nature of our systems is that we treat the semantic-parser learner or the language-generator learner as blackboxes. Both Krisp and Wasp were designed with supervised training data in mind. To create a unified framework, it would make more sense to modify the learning algorithms for semantic parsing and NLG so that they can deal with ambiguous supervision directly.

### 6.1.2 Learning Higher-order Concepts

In the sportscasting problem, some statements in the commentaries specifically refer to a pattern of activity across several recent events rather than to a single event. For example, in one of the English commentaries, the statement "Purple team is very sloppy today." appears after a series of turn-overs to the other team. The simulated perception could be extended to extract patterns of activity such as "sloppiness"; however, this assumes that such concepts are predefined, and extracting many such higher-level predicates would greatly increase ambiguity in the training data. The sportscasting system assumes it already has concepts for the words it needs to learn and can perceive these concepts and represent them in the MRL. However, it would be interesting to include a more "Whorfian" style of language learning (Whorf, 1964) in which an unknown word such as "sloppiness" could actually cause the creation of a new concept. For content words that do not seem to consistently correlate with any perceived event, the system could collect examples of recent activity where the word is used and try to learn a new higher-level concept that captures a regularity in these situations. For example, given examples of situations referred to as "sloppy," an inductive logic programming system (Lavrač & Džeroski, 1994) should be able to detect the pattern of several recent turnovers.

The ability to automatically construct higher level abstractions will free us from building a representation that encompasses all possible things we want to talk about. Instead of anticipating all potential actions and events, we could keep the representation at a low level and let the system build new concepts as needed. In particular, in certain continuous domains, we would not have to build specific bins for possible ranges of value. Instead, the system can infer how to categorize these ranges automatically. For example, for a weather forecast application, the system can learn the correspondences between temperature ranges and descriptions such as *hot*, *warm*, *cool*, or *cold*.

The navigation system does learn some high-level instructions since it learns to map words to any connected subgraph. Thus, the lexicon learning step is free to discover new concepts that were not predefined. However, the different candidate meanings learned are listed as separate entires in the lexicon. Moreover, we use a greedy approach of picking the highest scoring entries and ignoring the rest. Ideally we would use a more probabilistic representation that incorporates all the different candidates into a single graph to allow more flexible interpretations of the meaning dependent on the actual context.

### 6.1.3 Incorporating Syntactic Information

Even though we have chosen to not rely on any syntactic information in our language learning process, for many practical applications we would have access to at least some of that information. Moreover, there are unsupervised approaches to syntax learning (Klein & Manning, 2004; Ponvert, Baldridge, &

Erk, 2011; Spitkovsky, Alshawi, Chang, & Jurafsky, 2011) that could help us learn the semantics of language. Being able to parse the sentences first could help the system determine which words are the content words and also how the different words interact with each other. This would make the grounding problem easier since we would have a better idea of which words might correspond to which semantic entities. Even basic information such as part of speech tags would bias the learning to map nouns to objects and verbs to actions in the world. Preliminary experiments using this idea have shown that it can help eliminate some bad entries in the lexicon.

### 6.1.4 Scaling the Systems

One of the issues we discussed while introducing the SGOLL algorithm in Section 4.4 is scaling the system. Since we use weaker supervision, we would expect to require more training data to achieve the same performance. Consequently, a system that learns from ambiguous supervision should be able to scale to larger datasets than a fully-supervised system to be competitive. The main bottlenecks in our systems currently are in the semantic-parser learning step. Building a fast, distributed system for learning semantic parsers would allow our systems to scale to larger datasets. Moreover, we would be able to incorporate more retraining loops to iteratively improve our estimation of the supervised training data and in turn train better semantic parsers.

## 6.2 Task-specific Improvements

Even though we have only used the sportscasting and navigation tasks as testbeds for our language grounding systems, there are many improvements that could be made if we were interested in building practical systems that would solve these particular problems. In this section, we look at specific changes we can make to enhance our existing systems in these two domains.

### 6.2.1 Sportscasting

Some of the limitations of the current sportscasting system are due to inadequate representation of the world. The event extraction system mostly only concentrates on who has possession of the ball. Some of the comments in our data, particularly in the Korean data, refer to information about events that are not currently represented in the extracted MRs. For example, a player dribbling the ball is not captured by our perceptual system. Thus, we could extend the event extractor to include more event types as well as more detailed information such as the length of a pass or the location of a player, etc.

In addition to including more events about what's happening on the field, it should also be noted that commentators do not always talk about the immediate actions that are happening. In typical, real sportscasts, there would be a play-by-play announcer that talks about the play on the field, and a color commentator that fills in the gaps when no interesting actions are happening. The color commentaries can refer to statistics about the game, background information, or analysis of the game. While some of these are difficult to obtain, it would be simple to augment the potential MRs to include events such as the current score or the number of turnovers, etc. Although these may be difficult to learn correctly, they potentially would make the commentaries much more natural and engaging.

From a purely entertainment point of view, good sportscasts are more about managing the emotions of the audience rather than reporting facts. Thus, to build a practical system, we would have to make sure the comments are not repetitive and go beyond just stating the facts about the games. For example, varying the voice depending on the game situation could make the audience feel more in sync with the actions in the game.

### 6.2.2 Navigation

For the navigation domain, we have only looked at the task of semantic parsing. However, it would also be useful to build a system that could generate navigation instructions such as for a GPS device. Since our work is focused on resolving referential ambiguity, there are no inherent limitations to extending our system to perform language generation as well. Similar to what we did for the sportscasting task, we could use the supervised training data we have estimated as input to a supervised learning algorithm such as WASP$^{-1}$ for training a language generator.

In terms of building a controller for mobile robots, the interface should allow more interactive dialogues. This allows the system to ask clarification questions and also allows the users to correct erroneous behaviors.

## 6.3 Data Acquisition

In Section 4.7 we have shown that data annotation can be quite easy when all we require the annotators to do is to use language in a relevant perceptual context. In the navigation domain, this amounted to gathering navigation instructions and follower traces. However, we randomly selected the routes for which to collect instructions. A natural extension would be to use active learning (Cohn, Atlas, & Ladner, 1994) to select more useful annotations. Since we work in virtual environments, we could set up the world any way we want. Consequently, we could manipulate the world in such a way as to allow the system to get instructions about actions or landmarks it is the least confident about.

We have thus far described how to collect data in an offline manner. However, it would make sense to build interactive systems that continually try to learn while they are in service. This could be done for both systems that do semantic parsing and NLG. For example, a mobile robot could ask for feedback from the human users indicating whether or not it is performing the correct actions. An instruction generating system could observe whether the user is behaving as intended. Beyond just reinforcements that indicate whether the previous actions or instructions were good or bad, the system could also additionally ask the users for demonstrations of good behaviors. In this "benevolent teacher" setting, the user is incentivized to improve the system because they would also benefit from a better system.

## 6.4 Real Perceptions

So far we have only dealt with virtual environments for both the sportscasting and the navigation tasks. An obvious extension to the sportscasting task is to apply it to *real* RoboCup games or real soccer games rather than simulated ones. Recent work by Rozinat, Zickler, Veloso, van der Aalst, and McMillen (2008) analyzed games in the RoboCup Small Size League using videos from the overhead camera. By using the symbolic event trace extracted by such perceptual system, our methods could be applied to real-world games. Using speech recognition to accept spoken language input is another obvious extension.

For the navigation domain, we could build a real robot to navigate in a real environment. We would need to use an object recognition system to help us identify landmarks in the environment. Once the world has been processed into various tokens of landmarks, we could then use our system to ground the natural language. Here the ability to recognize patterns and automatically build higher level representations will be even more useful. Instead of predefining what tokens to learn and use, we can directly process the low level vector representation (e.g. SIFT descriptors of interest points (Lowe, 1999)) and automatically discover objects and the words that are used to describe that object.

## 6.5 Machine Translation

The idea behind WASP is to borrow method from statistical machine translation. Thus, it is only natural that we can apply the lessons we have learned back to the task of machine translation. We can use our algorithm to build translation models from ambiguously paired parallel corpora. This kind of data can occur when the translation is loose and not sentence-aligned. Thus, we can build a system that simultaneously aligns the sentences to each other and builds a translation model using an EM-like retraining loop. We could first train on sentences in one language matched to every sentence in the corresponding paragraph in the other language. Then we can use the translation model to help decide which sentences are the best matches. We can then iterate this process until convergence.

One import characteristic of the ambiguous problem we have examined is that the information expressed on the two sides are not equal. In particular, the NL sentences in our data only corresponds to a subset of the MRs we extract. This is similar to the translation scenario where one document is a summary of the other. For example, Wikipedia articles in other languages often contain only a subset of the information in the corresponding English article. Thus, deciding whether an alignment should even be made is important. We could use a method similar to our technique for superfluous comment removal to not align the least-likely sentence pairs. Since the ambiguity level can be high and many sentences may not have matches, we can also exploit token similarities or known word translations (e.g. the name of the wikipedia article) to reduce the number of possible alignments.

Another way to use our idea for machine translation is to do interlingual translation (Wong, 2007). The MRL can be used as an intermediate language for translating to any language. In other words, we would first parse the source sentence into its MR. Then, we would use the MR to generate sentences in the target language. For translating between $n$ languages, we only need $2n$ models instead of $\frac{n \cdot (n-1)}{2}$ models. Furthermore, we no longer need parallel corpora between the different languages. It is only necessary that they align to the same MRL. In other words, we are aligning the different languages using the world they describe instead of directly to each other. This can be a very useful application for the sportscasting domain where simultaneous broadcasts in many different languages are often required for large international sporting events such as the Olympics or the World Cup.

# Chapter 7

# Conclusion

Learning the semantics of language from the perceptual context in which it is uttered is a useful approach because only minimal human supervision is required. Instead of annotating each sentence with its correct semantic representation, the human teacher only has to demonstrate to the system how to use language in a relevant context. However, resolving the ambiguity of which parts of the perceptual context are being referred to can be a difficult problem. In this thesis, we have looked at a couple of frameworks aimed to solve this problem. The first system uses an EM-like retraining loop that alternates between building a semantic model of the language and estimating the mostly likely alignments between NL sentences and MRs. We demonstrated the feasibility of this system by applying it to a sportscasting task where the training data consist of textual commentaries and streams of automatically extracted events from simulated RoboCup games. We evaluated several scoring functions for disambiguating the training data in order to learn semantic parsers and language generators. Using a generation evaluation metric as the criterion for selecting the best NL–MR pairs produced better results than using semantic parsing scores when the initial training data were very noisy. Our system also learned a simple model for content selection from the ambiguous training data by estimating the probability that each event type evokes human commentary. Experimental evaluation verified that the overall system learned to accurately parse and generate comments as well as generate complete play-by-play sportscasts that are competitive with those produced by humans. We achieved similar results learning to sportscast in English and in Korean, demonstrating that our system is indeed language-independent and can adapt to learning new languages without modifications.

Given that this approach requires enumerating all the possible NL–MR alignments and scoring each one, we next look at another framework that can deal with more complicated data. In particular, we represent the space of candidates using a graph instead of a list and allow each NL sentence to potentially map to any of the exponential number of connected subgraphs. In addition to compactly representing the space of candidates, the graphical structure also allows us to encode the relationships between the different semantic entities. We use a lexicon-learning algorithm to first learn the meanings of words and short phrases, and then use the learned lexicon to prune the graph. This framework is applied to the problem of learning to interpret navigation instructions. The training data consist of textual navigation instructions and recorded action traces of people trying to follow those instructions. Since only the low-level actions are observed and not the actual navigation plans referred to in the instructions, we first construct the graph that represents the space of possible plans for each instruction. Experimental results showed that our system is able to infer the correct navigation plans from this space and correctly follow over half of the novel single-sentence instructions encountered during testing. Learning to follow Chinese navigation instructions produced similar results, again affirming the generality of our system. Finally, we looked at collecting more training data using Mechanical Turk. We showed that the task is indeed easy enough for anyone to annotate since they only have to understand how to give and take navigation instructions.

Learning natural language is a difficult and potentially never-ending task. As the world changes, the way we use language is also constantly evolving. Consequently, it is vital for a computer system to be able to continually adapt and learn from its experiences. Fortunately, the world is full of examples of how language is used. The difficulty, of course, is that these examples provide only very weak supervision. Here we have looked at one type of weak supervision in which the correct semantic annotations are assumed to be inside a finite hypothesis space. There are many other types of supervision as well. For example, the system might receive positive or negative reinforcements for a particular interpretation of language. Or the system could observe the results of using language (ask a question, issue a command, etc). Properly identifying and acquiring relevant data and utilizing the different kinds of supervision available are all keys to building a system could eventually solve the language-understanding problem.

# Appendices

# Appendix A

# Details of the Sportscasting Task

We use a rule-based system to automatically extract events from simulated RoboCup games. The type of events we detect are listed in Table A.1 along with a brief explanation of each event type.

| Event | Description |
|---|---|
| Playmode | The current play mode as defined by the game including kickoff, corner kick, goals, etc |
| Ballstopped | The ball speed has fallen below a minimum threshold |
| Turnover | The current possessor of the ball and the last possessor are on different teams |
| Kick | A player having possession of the ball in one time interval but not in the next |
| Pass | A player gains possession of the ball from a different player on the same team |
| BadPass | A pass in which the player gaining possession of the ball is on a different team |
| Defense | A turnover to the defensive team in their penalty area |
| Steal | A player having possession of the ball in one time interval and another player on a different team having it in the next time interval |
| Block | Turnover to the opposing goalie. |

Table A.1: Description of the different events detected

Below we include the context-free grammar we developed for our meaning representation language. All derivations start with the root symbol *S.

```
*S -> playmode ( *PLAYMODE )
*S -> ballstopped
*S -> turnover ( *PLAYER , *PLAYER )
*S -> kick ( *PLAYER )
*S -> pass ( *PLAYER , *PLAYER )
*S -> badPass ( *PLAYER , *PLAYER )
*S -> defense ( *PLAYER , *PLAYER )
*S -> steal ( *PLAYER )
*S -> block ( *PLAYER )

*PLAYMODE -> kick_off_l
*PLAYMODE -> kick_off_r
*PLAYMODE -> kick_in_l
*PLAYMODE -> kick_in_r
*PLAYMODE -> play_on
*PLAYMODE -> offside_l
*PLAYMODE -> offside_r
*PLAYMODE -> free_kick_l
*PLAYMODE -> free_kick_r
*PLAYMODE -> corner_kick_l
*PLAYMODE -> corner_kick_r
*PLAYMODE -> goal_kick_l
*PLAYMODE -> goal_kick_r
*PLAYMODE -> goal_l
*PLAYMODE -> goal_r

*PLAYER -> pink1
*PLAYER -> pink2
*PLAYER -> pink3
*PLAYER -> pink4
*PLAYER -> pink5
*PLAYER -> pink6
*PLAYER -> pink7
*PLAYER -> pink8
*PLAYER -> pink9
*PLAYER -> pink10
*PLAYER -> pink11
*PLAYER -> purple1
*PLAYER -> purple2
*PLAYER -> purple3
*PLAYER -> purple4
*PLAYER -> purple5
*PLAYER -> purple6
*PLAYER -> purple7
```

```
*PLAYER -> purple8
*PLAYER -> purple9
*PLAYER -> purple10
*PLAYER -> purple11
```

# Appendix B

# Details of the Navigation Task

The navigation task was originally designed by MacMahon et al. (2006) who collected the English instructions and the follower data that we use in our experiments. Below we show maps of the three virtual worlds as well as CFGs for the formal language we use to describe navigation plans. For more details about how the original task was designed, see MacMahon's (2007) Ph.D. thesis.

## B.1 Maps of the Virtual Worlds

MacMahon et al. (2006) constructed three virtual worlds for which they collected navigation instructions and follower data for. The three worlds are named Grid, L, and Jelly. The world Grid has the most compact design, forming a grid-like world. The world Jelly has the sparsest design, with the hallways spread out from each other. World L lies between those two in terms of compactness. Maps of the three worlds can be seen in Figures B.1, B.2, B.3[1].

[1]The map for the world Jelly has an extra concrete hallway compared to what was previously presented (MacMahon, 2007; Chen & Mooney, 2011). There were some discrepancies in the MARCO code and the documentations so we decided to just include it when we reconstructed the worlds for our data collection on Mechanical Turk.

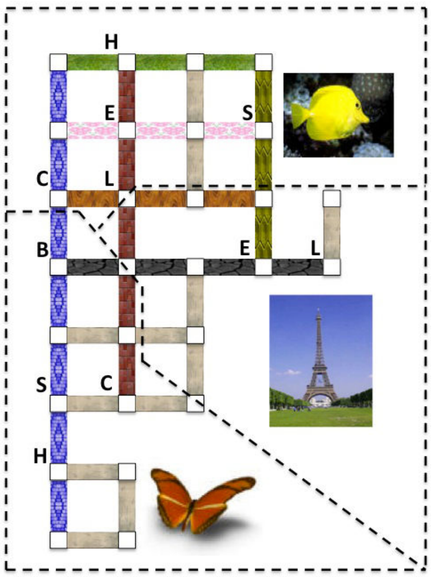


Figure B.1: Map of the virtual world Grid

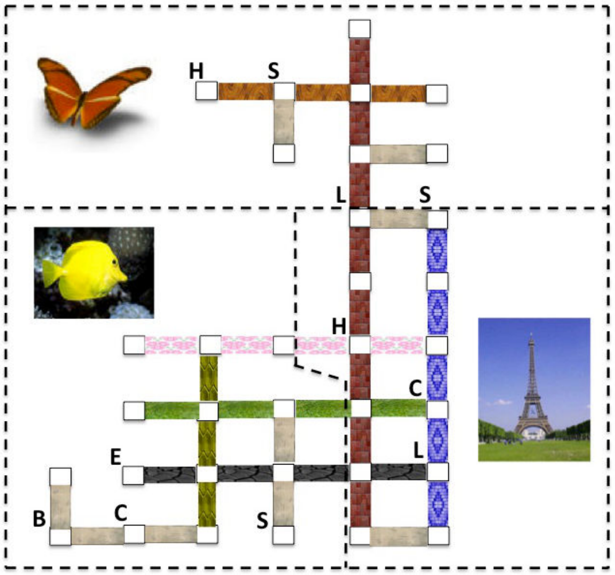


Figure B.2: Map of the virtual world L

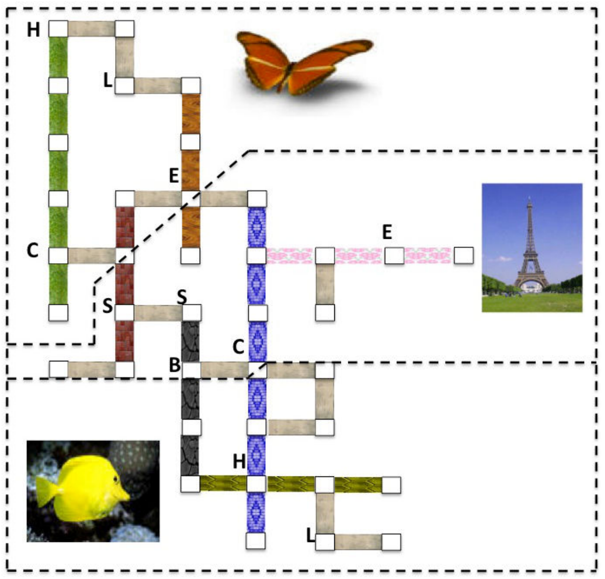


Figure B.3: Map of the virtual world Jelly

Each world consists of seven hallways with floor patterns and several short concrete hallways. The seven floor patterns are grass, brick, wood, gravel, blue, flower, and yellow octagons. In addition to the floor patterns, the world is divided into three areas, each with a different painting on the walls (butterfly, fish, and Eiffel Tower). There are also furnitures placed at various intersections (hatrack, lamp, chair, sofa, barstool, and easel). These are indicated on the maps with the first letter of their names.

## B.2 Meaning Representation Language

To facilitate learning, we did not use the navigation plan representations used by MacMahon et al. (2006) which were too complex and included many equivalent representations. Instead, we developed another formal language called the Steering Action Instruction Language (SAIL) that more easily captures the space of potential plans. All derivations start with the root symbol *S. *Num is a special non-terminal that can map to any integers.

The first grammar shown below is the initial CFG we developed for SAIL. It is compact and contains many recursive rules for describing arbitrarily long action sequences or list of arguments.

```
*S -> NULL
*S -> *Action

*Action -> *Action, *Action
*Action -> *Travel
*Action -> *Turn
*Action -> Verify( *Condition )

*Travel -> Travel( )
*Travel -> Travel( steps: *Num )

*Turn -> Turn( )
*Turn -> Turn( LEFT )
*Turn -> Turn( RIGHT )

*Condition -> *Condition, *Condition
*Condition -> *Direction *Object

*Direction -> at:
*Direction -> left:
*Direction -> right:
*Direction -> front:
*Direction -> back:
*Direction -> side:

*Object -> CHAIR
*Object -> BARSTOOL
*Object -> SOFA
*Object -> LAMP
*Object -> HATRACK
*Object -> EASEL
```

```
*Object -> 4-INTERSECTION
*Object -> 3-INTERSECTION
*Object -> *Wall
*Object -> *Hallway

*Wall -> WALL
*Wall -> *WallPaintings WALL

*WallPaintings -> FISH
*WallPaintings -> TOWER
*WallPaintings -> BUTTERFLY

*Hallway -> HALLWAY
*Hallway -> *HallwayTiles HALLWAY

*HallwayTiles -> GRASS
*HallwayTiles -> FLOWER
*HallwayTiles -> BLUE
*HallwayTiles -> WOOD
*HallwayTiles -> YELLOW
*HallwayTiles -> BRICK
*HallwayTiles -> GRAVEL
*HallwayTiles -> CONCRETE
```

As explained in Section 4.5.3, we developed another CFG for SAIL that corresponds better to natural language. This new grammar expands many of the production rules in the compact grammar.

```
*S -> NULL
*S -> Travel( )
*S -> Travel( steps: *Num )
*S -> Turn( )
*S -> Turn( LEFT )
*S -> Turn( RIGHT )
*S -> Verify( at: *Object )
*S -> Verify( left: *Object )
*S -> Verify( right: *Object )
*S -> Verify( front: *Object )
*S -> Verify( back: *Object )
*S -> Verify( side: *Object )
```

```
*S -> Verify( at: *Object, *Condition )
*S -> Verify( left: *Object, *Condition )
*S -> Verify( right: *Object, *Condition )
*S -> Verify( front: *Object, *Condition )
*S -> Verify( back: *Object, *Condition )
*S -> Verify( side: *Object, *Condition )
*S -> Verify( at: *Object ), Travel( )
*S -> Verify( left: *Object ), Travel( )
*S -> Verify( right: *Object ), Travel( )
*S -> Verify( front: *Object ), Travel( )
*S -> Verify( back: *Object ), Travel( )
*S -> Verify( side: *Object ), Travel( )
*S -> Verify( at: *Object, *Condition ), Travel( )
*S -> Verify( left: *Object, *Condition ), Travel( )
*S -> Verify( right: *Object, *Condition ), Travel( )
*S -> Verify( front: *Object, *Condition ), Travel( )
*S -> Verify( back: *Object, *Condition ), Travel( )
*S -> Verify( side: *Object, *Condition ), Travel( )
*S -> Verify( at: *Object ), Travel( steps: *Num )
*S -> Verify( left: *Object ), Travel( steps: *Num )
*S -> Verify( right: *Object ), Travel( steps: *Num )
*S -> Verify( front: *Object ), Travel( steps: *Num )
*S -> Verify( back: *Object ), Travel( steps: *Num )
*S -> Verify( side: *Object ), Travel( steps: *Num )
*S -> Verify( at: *Object, *Condition ), Travel( steps: *Num )
*S -> Verify( left: *Object, *Condition ), Travel( steps: *Num )
*S -> Verify( right: *Object, *Condition ), Travel( steps: *Num )
*S -> Verify( front: *Object, *Condition ), Travel( steps: *Num )
*S -> Verify( back: *Object, *Condition ), Travel( steps: *Num )
*S -> Verify( side: *Object, *Condition ), Travel( steps: *Num )
*S -> Verify( at: *Object ), Turn( )
*S -> Verify( left: *Object ), Turn( )
*S -> Verify( right: *Object ), Turn( )
*S -> Verify( front: *Object ), Turn( )
*S -> Verify( back: *Object ), Turn( )
*S -> Verify( side: *Object ), Turn( )
*S -> Verify( at: *Object, *Condition ), Turn( )
*S -> Verify( left: *Object, *Condition ), Turn( )
*S -> Verify( right: *Object, *Condition ), Turn( )
*S -> Verify( front: *Object, *Condition ), Turn( )
*S -> Verify( back: *Object, *Condition ), Turn( )
*S -> Verify( side: *Object, *Condition ), Turn( )
*S -> Verify( at: *Object ), Turn( LEFT )
*S -> Verify( left: *Object ), Turn( LEFT )
*S -> Verify( right: *Object ), Turn( LEFT )
```

```
*S -> Verify( front: *Object ), Turn( LEFT )
*S -> Verify( back: *Object ), Turn( LEFT )
*S -> Verify( side: *Object ), Turn( LEFT )
*S -> Verify( at: *Object, *Condition ), Turn( LEFT )
*S -> Verify( left: *Object, *Condition ), Turn( LEFT )
*S -> Verify( right: *Object, *Condition ), Turn( LEFT )
*S -> Verify( front: *Object, *Condition ), Turn( LEFT )
*S -> Verify( back: *Object, *Condition ), Turn( LEFT )
*S -> Verify( side: *Object, *Condition ), Turn( LEFT )
*S -> Verify( at: *Object ), Turn( RIGHT )
*S -> Verify( left: *Object ), Turn( RIGHT )
*S -> Verify( right: *Object ), Turn( RIGHT )
*S -> Verify( front: *Object ), Turn( RIGHT )
*S -> Verify( back: *Object ), Turn( RIGHT )
*S -> Verify( side: *Object ), Turn( RIGHT )
*S -> Verify( at: *Object, *Condition ), Turn( RIGHT )
*S -> Verify( left: *Object, *Condition ), Turn( RIGHT )
*S -> Verify( right: *Object, *Condition ), Turn( RIGHT )
*S -> Verify( front: *Object, *Condition ), Turn( RIGHT )
*S -> Verify( back: *Object, *Condition ), Turn( RIGHT )
*S -> Verify( side: *Object, *Condition ), Turn( RIGHT )

*S -> Travel( ), *Action
*S -> Travel( steps: *Num ), *Action
*S -> Turn( ), *Action
*S -> Turn( LEFT ), *Action
*S -> Turn( RIGHT ), *Action
*S -> Verify( at: *Object ), *Action
*S -> Verify( left: *Object ), *Action
*S -> Verify( right: *Object ), *Action
*S -> Verify( front: *Object ), *Action
*S -> Verify( back: *Object ), *Action
*S -> Verify( side: *Object ), *Action
*S -> Verify( at: *Object, *Condition ), *Action
*S -> Verify( left: *Object, *Condition ), *Action
*S -> Verify( right: *Object, *Condition ), *Action
*S -> Verify( front: *Object, *Condition ), *Action
*S -> Verify( back: *Object, *Condition ), *Action
*S -> Verify( side: *Object, *Condition ), *Action
*S -> Verify( at: *Object ), Travel( ), *Action
*S -> Verify( left: *Object ), Travel( ), *Action
*S -> Verify( right: *Object ), Travel( ), *Action
*S -> Verify( front: *Object ), Travel( ), *Action
*S -> Verify( back: *Object ), Travel( ), *Action
*S -> Verify( side: *Object ), Travel( ), *Action
```

```
*S -> Verify( at: *Object, *Condition ), Travel( ), *Action
*S -> Verify( left: *Object, *Condition ), Travel( ), *Action
*S -> Verify( right: *Object, *Condition ), Travel( ), *Action
*S -> Verify( front: *Object, *Condition ), Travel( ), *Action
*S -> Verify( back: *Object, *Condition ), Travel( ), *Action
*S -> Verify( side: *Object, *Condition ), Travel( ), *Action
*S -> Verify( at: *Object ), Travel( steps: *Num ), *Action
*S -> Verify( left: *Object ), Travel( steps: *Num ), *Action
*S -> Verify( right: *Object ), Travel( steps: *Num ), *Action
*S -> Verify( front: *Object ), Travel( steps: *Num ), *Action
*S -> Verify( back: *Object ), Travel( steps: *Num ), *Action
*S -> Verify( side: *Object ), Travel( steps: *Num ), *Action
*S -> Verify( at: *Object, *Condition ), Travel( steps: *Num ), *Action
*S -> Verify( left: *Object, *Condition ), Travel( steps: *Num ), *Action
*S -> Verify( right: *Object, *Condition ), Travel( steps: *Num ), *Action
*S -> Verify( front: *Object, *Condition ), Travel( steps: *Num ), *Action
*S -> Verify( back: *Object, *Condition ), Travel( steps: *Num ), *Action
*S -> Verify( side: *Object, *Condition ), Travel( steps: *Num ), *Action
*S -> Verify( at: *Object ), Turn( ), *Action
*S -> Verify( left: *Object ), Turn( ), *Action
*S -> Verify( right: *Object ), Turn( ), *Action
*S -> Verify( front: *Object ), Turn( ), *Action
*S -> Verify( back: *Object ), Turn( ), *Action
*S -> Verify( side: *Object ), Turn( ), *Action
*S -> Verify( at: *Object, *Condition ), Turn( ), *Action
*S -> Verify( left: *Object, *Condition ), Turn( ), *Action
*S -> Verify( right: *Object, *Condition ), Turn( ), *Action
*S -> Verify( front: *Object, *Condition ), Turn( ), *Action
*S -> Verify( back: *Object, *Condition ), Turn( ), *Action
*S -> Verify( side: *Object, *Condition ), Turn( ), *Action
*S -> Verify( at: *Object ), Turn( LEFT ), *Action
*S -> Verify( left: *Object ), Turn( LEFT ), *Action
*S -> Verify( right: *Object ), Turn( LEFT ), *Action
*S -> Verify( front: *Object ), Turn( LEFT ), *Action
*S -> Verify( back: *Object ), Turn( LEFT ), *Action
*S -> Verify( side: *Object ), Turn( LEFT ), *Action
*S -> Verify( at: *Object, *Condition ), Turn( LEFT ), *Action
*S -> Verify( left: *Object, *Condition ), Turn( LEFT ), *Action
*S -> Verify( right: *Object, *Condition ), Turn( LEFT ), *Action
*S -> Verify( front: *Object, *Condition ), Turn( LEFT ), *Action
*S -> Verify( back: *Object, *Condition ), Turn( LEFT ), *Action
*S -> Verify( side: *Object, *Condition ), Turn( LEFT ), *Action
*S -> Verify( at: *Object ), Turn( RIGHT ), *Action
*S -> Verify( left: *Object ), Turn( RIGHT ), *Action
*S -> Verify( right: *Object ), Turn( RIGHT ), *Action
```

```
*S -> Verify( front: *Object ), Turn( RIGHT ), *Action
*S -> Verify( back: *Object ), Turn( RIGHT ), *Action
*S -> Verify( side: *Object ), Turn( RIGHT ), *Action
*S -> Verify( at: *Object, *Condition ), Turn( RIGHT ), *Action
*S -> Verify( left: *Object, *Condition ), Turn( RIGHT ), *Action
*S -> Verify( right: *Object, *Condition ), Turn( RIGHT ), *Action
*S -> Verify( front: *Object, *Condition ), Turn( RIGHT ), *Action
*S -> Verify( back: *Object, *Condition ), Turn( RIGHT ), *Action
*S -> Verify( side: *Object, *Condition ), Turn( RIGHT ), *Action

*Action -> *Action, *Action
*Action -> Travel( )
*Action -> Travel( steps: *Num )
*Action -> Turn( )
*Action -> Turn( LEFT )
*Action -> Turn( RIGHT )
*Action -> Verify( at: *Object )
*Action -> Verify( left: *Object )
*Action -> Verify( right: *Object )
*Action -> Verify( front: *Object )
*Action -> Verify( back: *Object )
*Action -> Verify( side: *Object )
*Action -> Verify( at: *Object, *Condition )
*Action -> Verify( left: *Object, *Condition )
*Action -> Verify( right: *Object, *Condition )
*Action -> Verify( front: *Object, *Condition )
*Action -> Verify( back: *Object, *Condition )
*Action -> Verify( side: *Object, *Condition )

*Condition -> *Condition, *Condition
*Condition -> at: *Object
*Condition -> left: *Object
*Condition -> right: *Object
*Condition -> front: *Object
*Condition -> back: *Object
*Condition -> side: *Object

*Object -> CHAIR
*Object -> BARSTOOL
*Object -> SOFA
*Object -> LAMP
*Object -> HATRACK
*Object -> EASEL
*Object -> 4-INTERSECTION
*Object -> 3-INTERSECTION
```

```
*Object -> WALL
*Object -> FISH WALL
*Object -> TOWER WALL
*Object -> BUTTERFLY WALL

*Object -> HALLWAY
*Object -> GRASS HALLWAY
*Object -> FLOWER HALLWAY
*Object -> BLUE HALLWAY
*Object -> WOOD HALLWAY
*Object -> YELLOW HALLWAY
*Object -> BRICK HALLWAY
*Object -> GRAVEL HALLWAY
*Object -> CONCRETE HALLWAY
```

# Bibliography

Aho, A. V., & Ullman, J. D. (1972). *The Theory of Parsing, Translation, and Compiling*. Prentice Hall, Englewood Cliffs, NJ.

Alishahi, A., & Fazly, A. (2010). Integrating syntactic knowledge into a model of cross-situational word learning. In *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*.

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., & Weinert, R. (1991). The HCRC map task corpus. *Language and Speech*, *34*, 351–366.

André, E., Binsted, K., Tanaka-Ishii, K., Luke, S., Herzog, G., & Rist, T. (2000). Three RoboCup simulation league commentator systems. *AI Magazine*, *21*(1), 57–66.

Angeli, G., Liang, P., & Klein, D. (2010). A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*.

Artzi, Y., & Zettlemoyer, L. (2011). Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*.

Bailey, D., Feldman, J., Narayanan, S., & Lakoff, G. (1997). Modeling embodied lexical development. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society.*

Barnard, K., Duygulu, P., Forsyth, D., de Freitas, N., Blei, D. M., & Jordan, M. I. (2003). Matching words and pictures. *Journal of Machine Learning Research*, *3*, 1107–1135.

Barzilay, R., & Lapata, M. (2005). Collective content selection for concept-to-text generation. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-05).*

Barzilay, R., & Lee, L. (2002). Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02).*

Berg, T. L., Berg, A. C., Edwards, J., & Forsyth, D. A. (2004). Who's in the picture. In *Advances in Neural Information Processing Systems 17 (NIPS 2004).*

Bordes, A., Usunier, N., & Weston, J. (2010). Label ranking under ambiguous supervision for learning semantic correspondences. In *Proceedings of the 27th International Conference on Machine Learning (ICML-2010).*

Borschinger, B., Jones, B. K., & Johnson, M. (2011). Reducing grounded learning tasks to grammatical inference. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11).*

Branavan, S., Chen, H., Zettlemoyer, L. S., & Barzilay, R. (2009). Reinforcement learning for mapping instructions to actions. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP).*

Branavan, S., Silver, D., & Barzilay, R. (2011). Learning to win by reading manuals in a monte-carlo framework. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-11).*

Branavan, S., Zettlemoyer, L., & Barzilay, R. (2010). Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10).*

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., & Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, *16*(2), 79–85.

Brown, P. F., Della Pietra, V. J., Della Pietra, S. A., & Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, *19*(2), 263–312.

Bruni, E., Tran, G., & Baroni, M. (2011). Distributional semantics from text and images. In *Proceedings of the EMNLP 2011 Geometrical Models for Natural Language Semantics (GEMS 2011) Workshop.*

Buehler, P., Everingham, M., & Zisserman, A. (2009). Learning sign language by watching TV (using weakly aligned subtitles). In *Proceedings of the 22nd IEEE Conference on Computer Vision and Pattern Recognition (CVPR-09).*

Bunescu, R. C., & Mooney, R. J. (2005). Subsequence kernels for relation extraction. In Weiss, Y., Schölkopf, B., & Platt, J. (Eds.), *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, Vancouver, BC.

Byron, D., Koller, A., Striegnitz, K., Cassell, J., Dale, R., Moore, J., & Oberlander, J. (2009). Report on the first NLG challenge on Generating Instructions in Virtual Environments (GIVE). In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, Athens, Greece.

Chang, P.-C., Galley, M., & Manning, C. (2008). Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the ACL Third Workshop on Statistical Machine Translation.*

Chen, D. L., & Dolan, W. B. (2011). Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-2011)*, Portland, OR.

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

Chen, D. L., & Mooney, R. J. (2008). Learning to sportscast: A test of grounded language acquisition. In *Proceedings of 25th International Conference on Machine Learning (ICML-2008)*, Helsinki, Finland.

Chen, D. L., & Mooney, R. J. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-11).*

Chen, M., Foroughi, E., Heintz, F., Kapetanakis, S., Kostiadis, K., Kummeneje, J., Noda, I., Obst, O., Riley, P., Steffens, T., Wang, Y., & Yin, X. (2003). Users manual: RoboCup soccer server manual for soccer server version 7.07 and later.. Available at `http://sourceforge.net/projects/sserver/`.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pp. 263–270, Ann Arbor, MI.

Clarke, J., Goldwasser, D., Chang, M.-W., & Roth, D. (2010). Driving semantic parsing from the worlds response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL-2010)*, pp. 18–27.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, *15*(2), 201–221.

Collins, M. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 263–270, Philadelphia, PA.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*, 1–38.

Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of ARPA Workshop on Human Language Technology*, pp. 128–132, San Diego, CA.

Duboue, P. A., & McKeown, K. R. (2003). Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP-03)*, pp. 121–128.

Farhadi, A., Hejrati, M., Sadeghi, M. A., Young, P., Rashtchian, C., Hockenmaier, J., & Forsyth, D. (2010). Every picture tells a story: Generating sentences from images. In *Proceedings of the 11th European Conference on Computer Vision (ECCV-10).*

Fazly, A., Alishahi, A., & Stevenson, S. (2010). A probabilistic computational model of cross-situational word learning. *Cognitive Science*, *34*(6), 1017–1063.

Feng, Y., & Lapata, M. (2010). How many words is a picture worth? Automatic caption generation for news images. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*.

Fleischman, M., & Roy, D. (2007). Situated models of meaning for sports video retrieval. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT-07)*, Rochester, NY.

Fleischman, M., & Roy, D. (2008). Grounded language modeling for automatic speech recognition of sports video. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*.

Frank, M. C., Goodman, N. D., & Tenenbaum, J. B. (2009). Using speakers' referential intentions to model early cross-situational word learning. *Psychological Science*, *20*, 578–585.

Ge, R., & Mooney, R. J. (2005). A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pp. 9–16, Ann Arbor, MI.

Ge, R., & Mooney, R. J. (2009). Learning a compositional semantic parser using an existing syntactic parser. In *Joint Conference of the 47th Annual*

*Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP).*

Gold, K., & Scassellati, B. (2007). A robot that uses existing vocabulary to infer non-visual word meanings from observation. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07).*

Goldwasser, D., Reichart, R., Clarke, J., & Roth, D. (2011). Confidence driven unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-11).*

Golland, D., Liang, P., & Klein, D. (2010). A game-theoretic approach to generating spatial descriptions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10).*

Gorniak, P., & Roy, D. (2005). Speaking with your sidekick: Understanding situated speech in computer role playing games. In *Proceedings of the 4th Conference on Artificial Intelligence and Interactive Digital Entertainment*, Stanford, CA.

Gupta, S., & Mooney, R. (2009). Using closed captions to train activity recognizers that improve video retrieval. In *Proceedings of the CVPR-09 Workshop on Visual and Contextual Learning from Annotated Images and Videos (VCL)*, Miami, FL.

Hajishirzi, H., Hockenmaier, J., Muellar, E. T., & Amir, E. (2011). Reasoning in RoboCup soccer narratives. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI-2011).*

Harnad, S. (1990). The symbol grounding problem. *Physica D*, *42*, 335–346.

Herzog, G., & Wazinski, P. (1994). VIsual TRAnslator: Linking perceptions and natural language descriptions. *Artificial Intelligence Review*, *8*(2/3), 175–187.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the Tenth European Conference on Machine Learning (ECML-98)*, pp. 137–142, Berlin. Springer-Verlag.

Jurcicek, J., Gasic, M., Keizer, S., Mairesse, F., Thomson, B., & Young, S. (2009). Transformation-based learning for semantic parsing. In *Interspeech*, Brighton, UK.

Kate, R. J., & Mooney, R. J. (2006). Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL-06)*, pp. 913–920, Sydney, Australia.

Kate, R. J. (2008). Transforming meaning representation grammars to improve semantic parsing. In *Proceedings of the Twelfth Conference on*

*Computational Natural Language Learning (CoNLL-2008)*, pp. 33–40, Manchester, UK.

Kate, R. J., & Mooney, R. J. (2007). Learning language semantics from ambiguous supervision. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, pp. 895–900, Vancouver, Canada.

Kerr, W., Cohen, P. R., & Chang, Y.-H. (2008). Learning and playing in Wubble World. In *Proceedings of the Fourth Artificial Intelligence for Interactive Digital Entertainment Conference (AIIDE)*, Palo Alto, CA.

Kim, J., & Mooney, R. J. (2010). Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-10)*.

Klein, D., & Manning, C. D. (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pp. 479–486, Barcelona, Spain.

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

Knight, K., & Hatzivassiloglou, V. (1995). Two-level, many-paths generation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, pp. 252–260, Cambridge, MA.

Kollar, T., Tellex, S., Roy, D., & Roy, N. (2010). Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI).*

Kruijff, G.-J. M., Zender, H., Jensfelt, P., & Christensen, H. I. (2007). Situated dialogue and spatial organization: What, where... and why?. *International Journal of Advanced Robotic Systems*, *4*(2).

Kulkarni, G., Premraj, V., Dhar, S., Li, S., Choi, Y., Berg, A. C., & Berg, T. (2011). Baby talk: Understanding and generating image descriptions. In *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition (CVPR-11).*

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., & Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10).*

Lau, T., Drews, C., & Nichols, J. (2009). Interpreting written how-to instructions. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-2009).*

Lavrač, N., & Džeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood.

Li, S., Kulkarni, G., Berg, T., Berg, A., & Choi, Y. (2011). Composing simple image descriptions using web-scale n-grams. In *Proceedings of* *the Fifteenth Conference on Computational Natural Language Learning (CoNLL-2011).*

Liang, P., Jordan, M. I., & Klein, D. (2009). Learning semantic correspondences with less supervision. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP).*

Liang, P., Jordan, M. I., & Klein, D. (2011). Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-11).*

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, *2*, 419–444.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Vol. 2.

Lu, W., Ng, H. T., & Lee, W. S. (2009). Natural language generation with tree conditional random fields. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pp. 400–409.

Lu, W., Ng, H. T., Lee, W. S., & Zettlemoyer, L. S. (2008). A generative model for parsing natural language to meaning representations. In *Proceedings* *of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, Honolulu, HI.

MacMahon, M. (2007). *Following Natural Language Route Instructions*. Ph.D. thesis, Electrical and Computer Engineering Department, University of Texas at Austin.

MacMahon, M., Stankiewicz, B., & Kuipers, B. (2006). Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*.

MacWhinney, B. (2000). *The CHILDES project: Tools for analyzing talk* (Third edition). Lawrence Erlbaum Associates, Mahwah, NJ.

Matuszek, C., Fox, D., & Koscher, K. (2010). Following directions using statistical machine translation. In *Proceedings of the 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*.

McKeown, K. R. (1985). Discourse strategies for generating natural-language text. *Artificial Intelligence*, *27*(1), 1–41.

Meza-Ruiz, I. V., Riedel, S., & Lemon, O. (2008). Spoken language understanding in dialogue systems, using a 2-layer Markov logic network: improving semantic accuracy. In *Proceedings of Londial*.

Mooney, R. J. (2007). Learning for semantic parsing. In Gelbukh, A. (Ed.), *Computational Linguistics and Intelligent Text Processing: Proceedings* *of the 8th International Conference, CICLing 2007, Mexico City*, pp. 311–324. Springer Verlag, Berlin.

Morsillo, N., Pal, C., & Nelson, R. (2009). Semi-supervised learning of visual classifiers from web images and text. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-2009).*

Och, F. J., & Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, *29*(1), 19–51.

Orkin, J., & Roy, D. (2009). Automatic learning and generation of social behavior from collective human gameplay. In *The Eighth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-09).*

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 311–318, Philadelphia, PA.

Ponvert, E., Baldridge, J., & Erk, K. (2011). Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-11).*

Poon, H., & Domingos, P. (2009). Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09).*

Poon, H., & Domingos, P. (2010). Unsupervised ontology induction from text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*.

Qu, S., & Chai, J. Y. (2010). Context-based word acquisition for situated dialogue in a virtual world. *Journal of Artificial Intelligence Research*, *37*, 242–277.

Quine, W. v. (1960). *Word and Object*. MIT Press.

Reckman, H., Orkin, J., & Roy, D. (2010). Learning meanings of words and constructions, grounded in a virtual game. In *Proceedings of the 10th Conference on Natural Language Processing (KONVENS-10)*.

Regier, T., & Carlson, L. A. (2001). Grounding spatial language in perception: An empirical and computational investigation. *Journal of Experimental Psychology: General*, *130*(2), 273–298.

Riezler, S., Prescher, D., Kuhn, J., & Johnson, M. (2000). Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, pp. 480–487, Hong Kong.

Roy, D. (2002). Learning visually grounded words and syntax for a scene description task. *Computer Speech and Language*, *16*(3), 353–385.

Roy, D. (2005). Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence*, *167*, 170–205.

Rozinat, A., Zickler, S., Veloso, M., van der Aalst, W., & McMillen, C. (2008). Analyzing multi-agent activity logs using process mining techniques. In *Proceedings of the 9th International Symposium on Distributed Autonomous Robotic Systems (DARS-08)*, Tsukuba, Japan.

Satoh, S., Nakamura, Y., & Kanade, T. (1997). Name-it: Naming and detecting faces in video by the integration of image and natural language processing. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Shieber, S. M. (1988). A uniform architecture for parsing and generation. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING-88)*, pp. 614–619, Budapest, Hungary.

Shimizu, N., & Haas, A. (2009). Learning to follow navigational route instructions. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-2009)*.

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

Snow, R., O'Connor, B., Jurafsky, D., & Ng, A. Y. (2008). Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*.

Snyder, B., & Barzilay, R. (2007). Database-text alignment via structured multilabel classification. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-2007).*

Spitkovsky, V. I., Alshawi, H., Chang, A. X., & Jurafsky, D. (2011). Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11).*

Srihari, R. K., & Burhans, D. T. (1994). Visual semantics: Extracting visual information from text accompanying pictures. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94).*

Stolcke, A. (1995). An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, *21*(2), 165–201.

Tellex, S., Kollar, T., Dickerson, S., Walter, M. R., Banerjee, A. G., Teller, S., & Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-11).*

Tellex, S., & Roy, D. (2009). Grounding spatial prepositions for video search. In *Proceedings of the Eleventh International Conference on Multimodal Interfaces (ICMI-09).*

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

Vogel, A., & Jurafsky, D. (2010). Learning to follow navigational directions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10).*

Whorf, B. L. (1964). *Language, Thought, and Reality: Selected Writings.* MIT Press.

Wong, Y. W. (2007). *Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques.* Ph.D. thesis, Department of Computer Sciences, University of Texas.

Wong, Y., & Mooney, R. J. (2006). Learning for semantic parsing with statistical machine translation. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL-06)*, pp. 439–446, New York City, NY.

Wong, Y., & Mooney, R. J. (2007). Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT-07)*, pp. 172–179, Rochester, NY.

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

Yamada, K., & Knight, K. (2001). A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001)*, pp. 523–530, Toulouse, France.

Yu, C., & Ballard, D. H. (2004). On the integration of grounding language and learning objects. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pp. 488–493.

Zaragoza, H., & Li, C.-H. (2005). Learning what to talk about in descriptive games. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-05)*, pp. 291–298, Vancouver, Canada.

Zelenko, D., Aone, C., & Richardella, A. (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research*, *3*, 1083–1106.

Zettlemoyer, L. S., & Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of 21st Conference on Uncertainty in Artificial Intelligence (UAI-2005)*, Edinburgh, Scotland.

Zettlemoyer, L. S., & Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Compu-*

*tational Natural Language Learning (EMNLP-CoNLL-07)*, pp. 678–687, Prague, Czech Republic.

Zettlemoyer, L. S., & Collins, M. (2009). Learning context-dependent mappings from sentences to logical form. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*.

# Vita

David Lieh-Chiang Chen was born in 1982 in Taipei City, the capital of Taiwan. After immigrating to the United States at the age of 12, he attended Presidio Middle School and Lowell High School, both located in San Francisco, California. After high school, David went on to study Electrical Engineering and Computer Science at the University of California at Berkeley where he received his Bachelor of Science degree in 2004. He then obtained his Masters of Science degree in Computer Science at the University of California at Los Angeles in 2006. Since then, he has been pursuing his doctoral degree at the University of Texas at Austin where he has been exploring his interests in natural language processing, machine learning, computer vision, robotics, and crowdsourcing. After graduation, he will join Google in Mountain View starting in June, 2012.

Permanent address: davidlchen@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.