The Dissertation Committee for Yuk Wah Wong
certifies that this is the approved version of the following dissertation:

Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques

Committee:

Raymond J. Mooney, Supervisor

Jason M. Baldridge

Inderjit S. Dhillon

Kevin Knight

Benjamin J. Kuipers

Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques

by

Yuk Wah Wong, B.Sc. (Hons); 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

August 2007

To my loving family.

Acknowledgments

It is often said that doing a Ph.D. is like being left in the middle of the ocean and learning how to swim alone. But I am not alone. I am fortunate to have met many wonderful people who have made my learning experience possible.

First of all, I would like to thank my advisor, Ray Mooney, for his guidance throughout my graduate study. Knowledgeable and passionate about science, Ray is the best mentor that I could ever hope for. I especially appreciate his patience to let me grow as a researcher, and the freedom he gave me to explore new ideas. I will definitely miss our weekly meetings, which have always been intellectually stimulating.

I would also like to thank my thesis committee, Jason Baldridge, Inderjit Dhillon, Kevin Knight, and Ben Kuipers, for their invaluable feedback on my work. I am especially grateful to Kevin Knight for lending his expertise in machine translation and generation, providing detailed comments on my manuscripts, and for taking the time to visit Austin for my defense.

As for my collaborators at UT, I would like to thank Rohit Kate and Ruifang Ge for co-developing some of the resources on which this research is based, including the ROBOCUP corpus. Greg Kuhlmann also deserves thanks for annotating the ROBOCUP corpus, as do Amol Nayate, Nalini Belaramani, Tess Martin and Hollie Baker for helping with the evaluation of my NLG systems.

I am very lucky to be surrounded by a group of highly motivated, energetic, and intelligent colleagues at UT, including Sugato Basu, Prem Melville, Misha

Bilenko and Tuyen Huynh in the Machine Learning group, and Katrin Erk, Pascal Denis and Alexis Palmer in the Computational Linguistics group. In particular, I would like to thank my officemates, Razvan Bunescu and Lily Mihalkova, and Jason Chaw from the Knowledge Systems group for being wonderful listeners during my most difficult year.

I will cherish the friendships that I formed here. I am particularly grateful to Peter Stone and Umberto Gabbi for keeping my passion for music alive.

My Ph.D. journey would not be possible without the unconditional support of my family. I would not be where I am today without their guidance and trust. For this I would like to express my deepest gratitude. Last but not least, I thank my fiancée Tess Martin for her companionship. She has made my life complete.

The research described in this thesis was supported by the University of Texas MCD Fellowship, Defense Advanced Research Projects Agency under grant HR0011-04-1-0007, and a gift from Google Inc.

YUK WAH WONG

The University of Texas at Austin
August 2007

Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques

Publication No. ____________________

Yuk Wah Wong, Ph.D.
The University of Texas at Austin, 2007

Supervisor: Raymond J. Mooney

One of the main goals of natural language processing (NLP) is to build automated systems that can understand and generate human languages. This goal has so far remained elusive. Existing hand-crafted systems can provide in-depth analysis of domain sub-languages, but are often notoriously fragile and costly to build. Existing machine-learned systems are considerably more robust, but are limited to relatively shallow NLP tasks.

In this thesis, we present novel statistical methods for robust natural language understanding and generation. We focus on two important sub-tasks, *semantic parsing* and *tactical generation*. The key idea is that both tasks can be treated as the translation between natural languages and formal meaning representation languages, and therefore, can be performed using state-of-the-art statistical machine translation techniques. Specifically, we use a technique called synchronous parsing, which has been extensively used in syntax-based machine translation, as the unifying framework for semantic parsing and tactical generation. The parsing and generation algorithms learn all of their linguistic knowledge from annotated corpora, and can handle natural-language sentences that are conceptually complex.

A nice feature of our algorithms is that the semantic parsers and tactical generators share the same learned synchronous grammars. Moreover, charts are used as the unifying language-processing architecture for efficient parsing and generation. Therefore, the generators are said to be the *inverse* of the parsers, an elegant property that has been widely advocated. Furthermore, we show that our parsers and generators can handle formal meaning representation languages containing logical variables, including predicate logic.

Our basic semantic parsing algorithm is called WASP. Most of the other parsing and generation algorithms presented in this thesis are extensions of WASP or its inverse. We demonstrate the effectiveness of our parsing and generation algorithms by performing experiments in two real-world, restricted domains. Experimental results show that our algorithms are more robust and accurate than the currently best systems that require similar supervision. Our work is also the first attempt to use the same *automatically-learned* grammar for both parsing and generation. Unlike previous systems that require manually-constructed grammars and lexicons, our systems require much less knowledge engineering and can be easily ported to other languages and domains.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

An indicator of machine intelligence is the ability to converse in human languages (Turing, 1950). One of the main goals of natural language processing (NLP) as a sub-field of artificial intelligence is to build automated systems that can understand and generate human languages. This goal has so far remained elusive. Manually-constructed knowledge-based systems can understand and generate domain sub-languages, but are notoriously fragile and costly to build. Statistical methods are considerably more robust, but are limited to relatively shallow NLP tasks such as part-of-speech tagging, syntactic parsing, and word sense disambiguation. Robust, broad-coverage NLP systems that are capable of understanding and generating human languages are still beyond reach.

Recent advances in information retrieval seem to suggest that automated systems can appear to be intelligent without any deep understanding of human languages. However, the success of Internet search engines critically depends on the redundancy of natural language expressions in Web documents. For example, given the following search query:

Why do radio stations' names start with W?

Google returns a link to the following Web document that contains the relevant information:[1]

[1]The search was performed in July 2007. URL of Google: `http://www.google.com/`

> Answer “**Why do** us eastern **radio station names start with W** except KDKA KYW and KQV and western **station names start** with K except WIBW and WHO?”...

Note that this document contains an expression that is almost identical to the search query. In contrast, when given rare queries such as:

> *Does Germany border China?*

search engines such as Google would have difficulty finding Web documents that contain the search query. This leads to poor search results:

> The Break-up of Communism in East **Germany** and Eastern Europe. ...
> Kuo **does** not, however, provide a comprehensive treatment of **China**’s...

To answer this query would require spatial reasoning, which is impossible unless the query is correctly understood.

Similar arguments can be made for other NLP tasks such as machine translation, which is the translation between natural languages. Current statistical machine translation systems typically depend on the redundancy of translation pairs in the training corpora. When given rare sentences such as *Does Germany border China?*, machine translation systems would have difficulty composing good translations for them. Such reliance on redundancy may be reduced by using meaning representations that are more compact than natural languages. This would require the machine translators being able to understand the source language as well as generate the target language.

In this thesis, we will present novel statistical methods for robust natural language understanding and generation. We will focus on two important sub-tasks, *semantic parsing* and *tactical generation*.

1.1 Semantic Parsing

Semantic parsing is the task of transforming natural-language sentences into complete, formal, symbolic *meaning representations* (MR) suitable for automated reasoning or further processing. It is an integral part of natural language interfaces to databases (Androutsopoulos et al., 1995). For example, in the GEOQUERY database (Zelle and Mooney, 1996), a semantic parser is used to transform natural language queries into formal queries. Below is a sample English query, and its corresponding Prolog logical form:

What is the smallest state by area?

answer(x_1,smallest(x_2,(state(x_1),area(x_1,x_2))))

This Prolog logical form would be used to retrieve an answer to the English query from the GEOQUERY database. Other potential uses of semantic parsing include machine translation (Nyberg and Mitamura, 1992), document summarization (Mani, 2001), question answering (Friedland et al., 2004), command and control (Simmons et al., 2003), and interfaces to advice-taking agents (Kuhlmann et al., 2004).

1.2 Natural Language Generation

Natural language generation is the task of constructing natural-language sentences from computer-internal representations of information. It can be divided into two sub-tasks: (1) *strategic generation*, which decides what meanings to express, and (2) *tactical generation*, which generates natural-language expressions for those meanings. This thesis is focused on the latter task of tactical generation. One of the earliest motivating applications for natural language generation is machine translation (Yngve, 1962; Wilks, 1973). It is also an important component of dialog systems (Oh and Rudnicky, 2000) and automatic summarizers (Mani, 2001). For example, in the CMU Communicator travel planning system (Oh and Rudnicky, 2000), the input to the tactical generation component is a frame of attribute-value pairs:

$$\begin{bmatrix} \textit{act} & \textsc{query} \\ \textit{content} & \textsc{depart-time} \\ \textit{depart-city} & \text{New York} \end{bmatrix}$$

The output of the tactical generator would be a natural language sentence that expresses the meaning represented by the input frame:

What time would you like to leave New York?

1.3 Thesis Contributions

Much of the early research on semantic parsing and tactical generation was focused on hand-crafted knowledge-based systems that require tedious amounts of domain-specific knowledge engineering. As a result, these systems are often too brittle for general use, and cannot be easily ported to other application domains. In response to this, various machine learning approaches to semantic parsing and tactical generation have been proposed since the mid-1990's. Regarding these machine learning approaches, a few observations can be made:

1. Many of the statistical learning algorithms for semantic parsing are designed for simple domains in which sentences can be represented by a single semantic frame (e.g. Miller et al., 1996).

2. Other learning algorithms for semantic parsing that can handle complex sentences are based on inductive logic programming or deterministic parsing,

which lack the robustness that characterizes statistical learning (e.g. Zelle and Mooney, 1996).

3. While tactical generators enhanced with machine-learned components are generally more robust than their non-machine-learned counterparts, most, if not all, are still dependent on manually-constructed grammars and lexicons that are very difficult to maintain (e.g. Carroll and Oepen, 2005).

In this thesis, we present a number of novel statistical learning algorithms for semantic parsing and tactical generation. These algorithms automatically learn all of their linguistic knowledge from annotated corpora, and can handle natural-language sentences that are conceptually complex. The resulting parsers and generators are more robust and accurate than the currently best methods requiring similar supervision, based on experiments in four natural languages and in two real-world, restricted domains.

The key idea of this thesis is that both semantic parsing and tactical generation are treated as language translation tasks. In other words:

1. Semantic parsing can be defined as the translation from a natural language (NL) into a formal *meaning representation language* (MRL).

2. Tactical generation can be defined as the translation from a formal MRL into an NL.

Both tasks are performed using state-of-the-art statistical machine translation techniques. Specifically, we use a technique called *synchronous parsing*. Originally introduced by Aho and Ullman (1972) to model the translation between formal languages, synchronous parsing has recently been used to model the translation between NLs (Yamada and Knight, 2001; Chiang, 2005). We show that synchronous parsing can be used to model the translation between NLs and MRLs as well. Moreover, the resulting semantic parsers and tactical generators share the same learned synchronous grammars, and charts are used as the unifying language-processing architecture for efficient parsing and generation. Therefore, the generators are said to be the *inverse* of the parsers, an elegant property that has been noted by a number of researchers (e.g. Shieber, 1988).

In addition, we show that the synchronous parsing framework can handle a variety of formal MRLs. We present two sets of semantic parsing and tactical generation algorithms for different types of MRLs, one for MRLs that are variable-free, one for MRLs that contain logical variables, such as predicate logic. Both sets of algorithms are shown to be effective in their respective application domains.

1.4 Thesis Outline

Below is a summary of the remaining chapters of this thesis:

- In Chapter 2, we provide a brief overview of semantic parsing, natural language generation, statistical machine translation, and synchronous parsing. We also describe the application domains that will be considered in subsequent chapters.

- In Chapter 3, we describe how semantic parsing can be done using statistical machine translation. We present a semantic parsing algorithm called WASP, short for *Word Alignment-based Semantic Parsing*. This chapter is focused on variable-free MRLs.

- In Chapter 4, we extend the WASP semantic parsing algorithm to handle target MRLs with logical variables. The resulting algorithm is called λ-WASP.

- In Chapter 5, we describe how tactical generation can be done using statistical machine translation. We present results on using a recent phrase-based statistical machine translation system, PHARAOH (Koehn et al., 2003), for tactical generation. We also present WASP^{-1}, which is the inverse of the WASP semantic parser, and two hybrid systems, PHARAOH++ and WASP^{-1}++. Among the four systems, WASP^{-1}++ is shown to be provide the best overall performance. This chapter is focused on variable-free MRLs.

- In Chapter 6, we extend the WASP^{-1}++ tactical generation algorithm to handle source MRLs with logical variables. The resulting algorithm is called λ-WASP^{-1}++.

- In Chapter 7, we show some preliminary results for *interlingual machine translation*, an approach to machine translation that integrates natural language understanding and generation. We also discuss the prospect of natural language understanding and generation for unrestricted texts, and suggest several possible future research directions toward this goal.

- In Chapter 8, we conclude this thesis.

Figure 1.1 summarizes the various algorithms presented in this thesis.

Some of the work presented in this thesis has been previously published. Material presented in Chapters 3, 4 and 5 appeared in Wong and Mooney (2006), Wong and Mooney (2007b) and Wong and Mooney (2007a), respectively.

	Variable-free MRLs	MRLs with logical variables
Semantic parsing	WASP *(Chapter 3)*	λ-WASP *(Chapter 4)*
Tactical generation	PHARAOH WASP^{-1} PHARAOH++ WASP^{-1}++ *(Chapter 5)*	λ-WASP^{-1}++ *(Chapter 6)*

Figure 1.1: The parsing and generation algorthms presented in this thesis

Chapter 2

Background

This thesis encompasses several areas of NLP: semantic parsing (or natural language understanding), natural language generation, and machine translation. These areas have traditionally formed separate research communities, to some degree isolated from each other. In this chapter, we provide a brief overview of these three areas of research. We also provide background on synchronous parsing and synchronous grammars, which we claim can form a unifying framework for these NLP tasks.

2.1 Application Domains

First of all, we review the application domains that will be considered in subsequent sections. Our main focus is on application domains that have been used for evaluating semantic parsers. These domains will be re-used for evaluating tactical generators (Section 5.2) and interlingual machine translation systems (Section 7.1).

Much work on learning for semantic parsing has been done in the context of spoken language understanding (SLU) (Wang et al., 2005). Among the application domains developed for benchmarking SLU systems, the ATIS (Air Travel Information Services) domain is probably the most well-known (Price, 1990). The ATIS corpus consists of spoken queries that were elicited by presenting human subjects with various hypothetical travel planning scenarios to solve. The resulting spontaneous spoken queries were recorded as the subjects interacted with automated dialog systems to solve the scenarios. The recorded speech was transcribed and annotated with SQL queries and reference answers. Below is a sample transcribed query with its SQL annotation:

Show me flights from Boston to New York.

```
SELECT filght_id FROM flight WHERE
     from_airport = 'boston'
     AND to_airport = 'new york'
```

The ATIS corpus exhibits a wide range of interesting phenomena often associated with spontaneous speech, such as verbal deletion and flexible word order. However, we will *not* focus on this domain in this thesis, because the SQL annotations tend to be quite messy, and it takes a lot of human effort to transform the SQL annotations into a usable form.[1] Also most ATIS queries are in fact conceptually very simple, and semantic parsing often amounts to slot filling of a single semantic frame (Kuhn and De Mori, 1995; Popescu et al., 2004). We mention this domain because much of the existing work described in Section 2.2 was developed for the ATIS domain.

In this thesis, we focus on the following two domains. The first one is GEOQUERY. The aim of this domain is to develop an NL interface to a U.S. geography database written in Prolog. This database was part of the Turbo Prolog 2.0 distribution (Borland International, 1988). The query language is basically first-order Prolog logical forms, augmented with several *meta-predicates* for dealing

[1]None of the existing ATIS systems that we are aware of use SQL directly. Instead, they use intermediate languages such as predicate logic (Zettlemoyer and Collins, 2007) which are then translated into SQL using external tools.

with quantification (Zelle and Mooney, 1996). The GEOQUERY corpus consists of written English, Spanish, Japanese and Turkish queries gathered from various sources. All queries were annotated with Prolog logical forms. Below is a sample English query and its Prolog annotation:

What states does the Ohio run through?

answer(x_1,(state(x_1),traverse(x_2,x_1),
    equal(x_2,riverid(ohio))))

Note that the *logical variables* x_1 and x_2 are used to denote entities. In this logical form, `state` is a predicate that returns true if its argument (x_1) denotes a U.S. state, and `traverse` is a predicate that returns true if its first argument (x_2), which is a river, traverses its second argument (x_1), which is usually a state. The `equal` predicate returns true if its first argument (x_2) denotes the Ohio river (`riverid(ohio)`). Finally, the logical variable x_1 denotes the answer (`answer`) to the query. In this domain, queries typically show a deeply nested structure, which makes the semantic parsing task rather challenging, e.g.:

What states border the states that the Ohio runs through?
What states border the state that borders the most states?

For semantic parsers that cannot deal with logical variables (e.g. Ge and Mooney, 2006; Kate and Mooney, 2006), a functional, *variable-free* query language (FUNQL) has been developed for this domain (Kate et al., 2005). In FUNQL, each predicate can be seen to have a set-theoretic interpretation. For example, in the FUNQL equivalent of the Prolog logical form shown above:

```
answer(state(traverse_1(riverid(ohio))))
```

the term `river(ohio)` denotes a singleton set that consists of the Ohio river, `traverse_1` denotes the set of entities that some of the members of its argument (which are rivers) run through[2], and `state` denotes the subset of its argument whose members are also U.S. states.

The second domain that we consider is ROBOCUP. ROBOCUP (`http://www.robocup.org/`) is an international AI research initiative that uses robotic soccer as its primary domain. In the ROBOCUP Coach Competition, teams of autonomous agents compete on a simulated soccer field, receiving advice from a team coach using a formal language called CLANG (Chen et al., 2003). Our specific aim is to develop an NL interface for autonomous agents to understand NL advice. The ROBOCUP corpus consists of formal CLANG advice mined from previous Coach Competition game logs, annotated with English translations. Below is a piece of CLANG advice and its English gloss:

```
((bowner our {4})
 (do our {6} (pos (left (half our)))))
```

If our player 4 has the ball, then our player 6 should stay in the left side of our half.

In CLANG, tactics are generally expressed in the form of if-then rules. Here the expression `(bowner ...)` represents the "ball owner" condition, and `(do ...)` is a directive that is followed when the condition holds, i.e. player 6 should position itself (`pos`) in the left side (`left`) of our half (`(half our)`).

Appendix A provides detailed specifications of all formal meaning represenation languages (MRL) being considered: the GEOQUERY logical query language,

[2]On the other hand, `traverse_2` is the inverse of `traverse_1`, i.e. it denotes the set of rivers that run through some of the members of its argument (which are usually cities or U.S. states).

FunQL, and CLang.

2.2 Semantic Parsing

Semantic parsing is a research area with a long history. Many early semantic parsers are NL interfaces to databases, including Lunar (Woods et al., 1972), Chat-80 (Warren and Pereira, 1982), and Tina (Seneff, 1992). These NL interfaces are often hand-crafted for a particular database, and cannot be easily ported to other domains. Over the last decade, various data-driven approaches to semantic parsing have been proposed. These algorithms often produce semantic parsers that are more robust and accurate, and tend to be less application-specific than their hand-crafted counterparts. In this section, we provide a brief overview of these learning approaches.

2.2.1 Syntax-Based Approaches

One of the earliest data-driven approaches to semantic parsing is based on the idea of augmenting statistical syntactic parsers with semantic labels. Miller et al. (1994) propose the hierarchical Hidden Understanding Model (HUM) in which context-free grammar (CFG) rules are learned from an annotated corpus consisting of *augmented parse trees*. Figure 2.1 shows a sample augmented parse tree in the Atis domain. Here the non-terminal symbols Flight, Stop and City represent domain-specific concepts, while other non-terminal symbols such as NP (noun phrase) and VP (verb phrase) are syntactic categories. Given an input sentence, a parser based on a probabilistic recursive transition network is used to find the best augmented parse tree. This tree is then converted into a non-recursive semantic frame using a probabilistic semantic interpretation model (Miller et al., 1996).

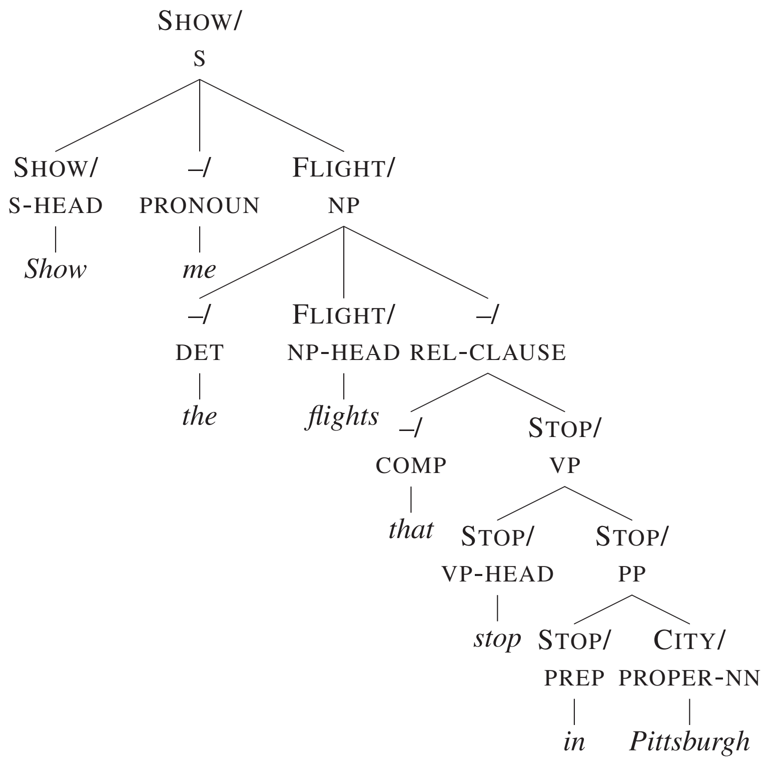


Figure 2.1: An augmented parse tree taken from Miller et al. (1994)

Ge and Mooney (2005, 2006) present another algorithm using augmented parse trees called SCISSOR. It is an improvement over HUM in three respects. First, it is based on a state-of-the-art statistical lexicalized parser (Bikel, 2004). Second, it handles meaning representations (MR) that are deeply nested, which are typical in the GEOQUERY and ROBOCUP domains. Third, a discriminative re-ranking model is used for incorporating non-local features. Again, training requires fully-annotated augmented parse trees.

The main drawback of HUM and SCISSOR is that they require augmented parse trees for training which are often very difficult to obtain. Zettlemoyer and Collins (2005) address this problem by treating parse trees as hidden variables which must be estimated using expectation-maximization (EM). Their method is based on a combinatory categorial grammar (CCG) (Steedman, 2000). The key idea is to first over-generate a CCG lexicon using a small set of language-specific template rules. For example, consider the following template rule:

Input trigger: any binary predicate p
Output category: $(\text{S}\backslash\text{NP})/\text{NP} : \lambda x_1.\lambda x_2.p(x_2, x_1)$

Suppose we are given a training sentence, *Utah borders Idaho*, and its logical form, `borders(utah,idaho)`. The binary predicate `borders` would trigger the above template rule, producing a lexical item for each word in the sentence:

Utah $:= (\text{S}\backslash\text{NP})/\text{NP} : \lambda x_1.\lambda x_2.\texttt{borders}(x_2, x_1)$
borders $:= (\text{S}\backslash\text{NP})/\text{NP} : \lambda x_1.\lambda x_2.\texttt{borders}(x_2, x_1)$
Idaho $:= (\text{S}\backslash\text{NP})/\text{NP} : \lambda x_1.\lambda x_2.\texttt{borders}(x_2, x_1)$

Next, spurious lexical items such as *Utah* and *Idaho* are pruned away during the parameter estimation phase, where log-linear parameters are learned. A later version of this work (Zettlemoyer and Collins, 2007) uses a relaxed CCG for dealing with flexible word order and other speech-related phenomena, as exemplified by the ATIS domain. Note that both CCG-based algorithms require prior knowledge of the NL syntax in the form of template rules for training.

2.2.2 Semantic Grammars

A common feature of syntax-based approaches is to generate full syntactic parse trees together with semantic parses. This is often a more elaborate structure than needed. One way to simplify the output is to remove syntactic labels from parse trees. This results in a *semantic grammar* (Allen, 1995), in which non-terminal symbols correspond to domain-specific concepts as opposed to syntactic categories. A sample semantic parse tree is shown in Figure 2.2.

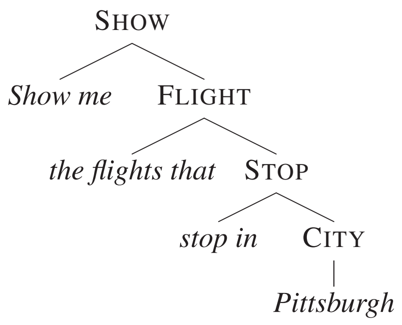


Figure 2.2: A semantic parse tree for the sentence in Figure 2.1

Several algorithms for learning semantic grammars have been devised. Kate et al. (2005) present a bottom-up learning algorithm called SILT. The key idea is to re-use the non-terminal symbols provided by a domain-specific MRL grammar (see Appendix A). Each production in the MRL grammar corresponds to a domain-specific concept. Given a training set consisting of NL sentences and their correct MRs, context-free parsing rules are learned for each concept, starting with rules that appear in the leaves of a semantic parse (e.g. CITY → *Pittsburgh*), followed by rules that appear one level higher (e.g. STOP → *stop in* CITY), and so on. The result is a semantic grammar that covers the training set.

More recently, Kate and Mooney (2006) present an algorithm called KRISP based on string kernels. Instead of learning individual context-free parsing rules for each domain-specific concept, KRISP learns a support vector machine (SVM) classifier with string kernels (Lodhi et al., 2002). The kernel-based classifier essentially assigns weights to all possible word subsequences up to a certain length, so that subsequences correlated with the specific concept receive higher weights. The learned model is thus equivalent to a weighted semantic grammar with many context-free parsing rules. It is shown that KRISP is more robust than other semantic parsers in the face of noisy input sentences.

In Chapters 3 and 4, we will introduce two semantic parsing algorithms, WASP and λ-WASP, which learn semantic grammars from annotated corpora using statistical machine translation techniques.

2.2.3 Other Approaches

Various other learning approaches have been proposed for semantic parsing. Kuhn and De Mori (1995) introduce a system called CHANEL that translates NL queries into SQL based on classifications given by learned decision trees. Each decision tree decides whether to include a particular attribute or constraint in the output SQL query. CHANEL has been deployed in the ATIS domain where queries are often conceptually simple.

Zelle and Mooney (1996) present a system called CHILL which is based on inductive logic programming (ILP). It learns a deterministic shift-reduce parser from an annotated corpus given a bilingual lexicon, which can be either handcrafted or automatically acquired (Thompson and Mooney, 1999). COCKTAIL (Tang and Mooney, 2001) is an extension of CHILL that shows better coverage through the use of multiple clause constructors.

Papineni et al. (1997) and Macherey et al. (2001) are two semantic parsing algorithms using machine translation. Both algorithms translate English ATIS queries into formal queries as if the target language were a natural language. Papineni et al. (1997) is based on a discriminatively-trained, word-based translation model (Section 2.5.1), while Macherey et al. (2001) is based on a phrase-based translation model (Section 2.5.2). Unlike these algorithms, our WASP and λ-WASP algorithms are based on syntax-based translation models (Section 2.5.2).

He and Young (2003, 2006) propose the Hidden Vector State (HVS) model, which is an extension of the hidden Markov model (HMM) with stack-oriented state vectors. It can capture the hierarchical structure of sentences, while being more constrained than CFGs. It has been deployed in various SLU systems including ATIS, and is shown to be quite robust to input noise.

Wang and Acero (2003) propose an extended HMM model for the ATIS domain, where a multiple-word segment is generated from each underyling Markov state that corresponds to a domain-specific semantic slot. These segments correspond to slot fillers such as dates and times, for which CFGs are written. Then a learned HMM serves to glue together different slot fillers to form a complete semantic interpretation.

Lastly, PRECISE (Popescu et al., 2003, 2004) is a knowledge-intensive approach to semantic parsing that does *not* involve any learning. It introduces the notion of *semantically tractable* sentences, sentences that give rise to a unique semantic interpretation given a hand-crafted lexicon and a set of semantic constraints. Interestingly, Popescu et al. (2004) shows that over 90% of the context-independent ATIS queries are semantically tractable, whereas only 80% of the GEOQUERY queries are semantically tractable, which shows that GEOQUERY is indeed a more challenging domain than ATIS.

Note that none of the above systems can be easily adapted for the inverse task of tactical generation. In Chapters 5 and 6, we will show that the WASP and λ-WASP semantic parsing algorithms (Chapters 3 and 4) can be readily inverted to produce effective tactical generators.

2.3 Natural Language Generation

This section provides a brief summary of data-driven approaches to natural language generation (NLG). More specifically, we focus on tactical generation, which is the generation of NL sentences from formal, symbolic MRs.

Early tactical generation systems, such as PENMAN (Bateman, 1990), SURGE (Elhadad and Robin, 1996), and REALPRO (Lavoie and Rambow, 1997), typically depend on large-scale knowledge bases that are built by hand. These systems are often too fragile for general use due to knowledge gaps in the hand-built grammars and lexicons.

To improve robustness, Knight and Hatzivassiloglou (1995) introduce a two-level architecture in which a statistical n-gram language model is used to rank the output of a knowledge-based generator. The reason for improved robustness is two-fold: First, when dealing with new constructions, the knowledge-based system can freely overgenerate, and let the language model make its selections. This simplifies the construction of knowledge bases. Second, when faced with incomplete or underspecified input (e.g. from semantic parsers), the language model can help fill in the missing pieces based on fluency.

Many subsequent NLG systems follow the same overall architecture. For example, NITROGEN (Langkilde and Knight, 1998) is an NLG system similar to Knight and Hatzivassiloglou (1995), but with a more efficient knowledge-based component that operates bottom-up rather than top-down. Again, a statistical n-gram ranker is used to extract the best output sentence from a set of candidates. HALOGEN (Langkilde-Geary, 2002) is a successor to NITROGEN, which includes a knowledge base that provides better coverage of English syntax.

FERGUS (Bangalore et al., 2000) is an NLG system based on the XTAG grammar (XTAG Research Group, 2001). Given an input dependency tree whose nodes are unordered and are labeled only with lexemes, a statistical tree model is used to assign the best elementary tree for each lexeme. Then a word lattice that encodes all possible surface strings permitted by the elementary trees is formed.

A trigram language model trained on the Wall Street Journal (WSJ) corpus is then used to rank the candidate strings.

AMALGAM (Corston-Oliver et al., 2002; Ringger et al., 2004) is an NLG system for French and German in which the mapping from underspecified to fully-specified dependency parses is mostly guided by learned decision tree classifiers. These classifiers insert function words, determine verb positions, re-attach nodes for raising and *wh*-movement, and so forth. These classifiers are trained on the output of hand-crafted, broad-coverage parsers. Hand-built classifiers are used whenever there is insufficient training data. A statistical language model is then used to determine the relative order of constituents in a dependency parse.

2.3.1 Chart Generation

The XTAG grammar used by FERGUS is a *bidirectional* (or *reversible*) grammar that has been used for parsing as well (Schabes and Joshi, 1988). The use of a single grammar for both parsing and generation has been widely advocated for its elegance. Kay's (1975) research into functional grammar is motivated by the desire to "make it possible to generate and analyze sentences with the same grammar". Jacobs (1985) presents an early implementation of this idea. His PHRED generator operates from the same declarative knowledge base used by PHRAN, a sentence analyzer (Wilensky and Arens, 1980). Other early NLP systems share at least part of the linguistic knowledge for parsing and generation (Steinacker and Buchberger, 1983; Wahlster et al., 1983).

Shieber (1988) notes that not only a single grammar can be used for parsing and generation, but also the same language-processing architecture can be used for processing the grammar in both directions. He suggests that *charts* can be a natural uniform architecture for efficient parsing and generation. This is in marked contrast to previous systems (e.g. PHRAN and PHRED) where the parsing and generation algorithms are often radically different. Kay (1996) further refines this idea, pointing out that chart generation is similar to chart parsing with free word order, because in logical forms, the relative order of predicates is immaterial.

These observations have led to the development of a number of chart generators. Carroll et al. (1999) introduce an efficient bottom-up chart generator for head-driven phrase structure grammars (HPSG). Constructions such as intersective modification (e.g. *a tall young Polish athlete*) are treated in a separate phase because chart generation can be exponential in these cases. Carroll and Oepen (2005) further introduce a procedure to selectively unpack a derivation forest based on a probabilistic model, which is a combination of a 4-gram language model and a maximum-entropy model whose feature types correspond to sub-trees of derivations (Velldal and Oepen, 2005).

White and Baldridge (2003) present a chart generator adapted for use with CCG. A major strength of the CCG generator is its ability to generate a wide range of coordination phenomena efficiently, including argument cluster coordination. A statisical n-gram language model is used to rank candidate surface strings (White, 2004).

Nakanishi et al. (2005) present a similar probabilistic chart generator based on the Enju grammar, an English HPSG grammar extracted from the Penn Treebank (Miyao et al., 2004). The probabilistic model is a log-linear model with a variety of n-gram features and syntactic features.

Despite their use of statistical models, all of the above algorithms rely on manually-constructed knowledge bases or grammars which are difficult to maintain. Moreover, they focus on the task of *surface realization*, i.e. linearizing and inflecting words in a sentence, requiring extensive lexical information (e.g. lexemes) in the input logical forms. The mapping from predicates to lexemes is then relegated to a separate sentence planning component. In Chapters 5 and 6, we will introduce tactical generation algorithms that learn all of their linguistic knowledge from annotated corpora, and show that surface realization and lexical selection can be integrated in an elegant framework based on synchronous parsing.

2.4 Synchronous Parsing

In this section, we define the notion of synchronous parsing. Originally introduced by Aho and Ullman (1969, 1972) to model the compilation of high-level programming languages into machine code, it has recently been used in various NLP tasks that involve language translation, such as machine translation (Wu, 1997; Yamada and Knight, 2001; Chiang, 2005; Galley et al., 2006), textual entailment (Wu, 2005), sentence compression (Galley and McKeown, 2007), question answering (Wang et al., 2007), and syntactic parsing for resource-poor languages (Chiang et al., 2006). Shieber and Schabes (1990a,b) propose that synchronous parsing can be used for semantic parsing and natural language generation as well.

Synchronous parsing differs from ordinary parsing in that a derivation yields a *pair* of strings (or trees). To finitely specify a potentially infinite set of string pairs (or tree pairs), we use a synchronous grammar. Many types of synchronous grammars have been proposed for NLP, including synchronous context-free grammars (Aho and Ullman, 1972), synchronous tree-adjoining grammars (Shieber and Schabes, 1990b), synchronous tree-substitution grammars (Yamada and Knight, 2001), and quasi-synchronous grammars (Smith and Eisner, 2006). In the next subsection, we will illustrate synchronous parsing using synchronous context-free grammars (SCFG).

2.4.1 Synchronous Context-Free Grammars

An SCFG is defined by a 5-tuple:

$$G = \langle \mathcal{N}, \mathcal{T}_e, \mathcal{T}_f, \mathcal{L}, S \rangle \tag{2.1}$$

where $\mathcal{N}$ is a finite set of non-terminal symbols, $\mathcal{T}_e$ is a finite set of terminal symbols for the *input language*, $\mathcal{T}_f$ is a finite set of terminal symbols for the *output language*, $\mathcal{L}$ is a *lexicon* consisting of a finite set of production rules, and $S \in \mathcal{N}$ is a distinguished start symbol. Each production rule in $\mathcal{L}$ takes the following form:

$$A \rightarrow \langle \alpha, \beta \rangle \tag{2.2}$$

where $A \in \mathcal{N}$, $\alpha \in (\mathcal{N} \cup \mathcal{T}_e)^+$, and $\beta \in (\mathcal{N} \cup \mathcal{T}_f)^+$. The non-terminal A is called the *left-hand side* (LHS) of the production rule. The *right-hand side* (RHS) of the production rule is a pair of strings, $\langle \alpha, \beta \rangle$. For each non-terminal in α, here is an *associated*, identical non-terminal in β. In other words, the non-terminals in α are a permutation of the non-terminals in β. We use indices $\boxed{1}, \boxed{2}, \ldots$ to indicate the association. For example, in the production rule $A \rightarrow \langle B_{\boxed{1}} B_{\boxed{2}}, B_{\boxed{2}} B_{\boxed{1}} \rangle$, the first B non-terminal in $B_{\boxed{1}} B_{\boxed{2}}$ is associated with the second B non-terminal in $B_{\boxed{2}} B_{\boxed{1}}$. Given an SCFG, G, we define a *translation form* as follows:

1. $\langle S_{\boxed{1}}, S_{\boxed{1}} \rangle$ is a translation form.

2. If $\langle \alpha A_{\boxed{i}} \beta, \alpha' A_{\boxed{i}} \beta' \rangle$ is a translation form, and if $A \rightarrow \langle \gamma, \gamma' \rangle$ is a production rule in $\mathcal{L}$, then $\langle \alpha\gamma\beta, \alpha'\gamma'\beta' \rangle$ is also a translation form. For this, we write:

   $$\langle \alpha A_{\boxed{i}} \beta, \alpha' A_{\boxed{i}} \beta' \rangle \Rightarrow_G \langle \alpha\gamma\beta, \alpha'\gamma'\beta' \rangle$$

   The non-terminals $A_{\boxed{i}}$ are said to be *rewritten* by the production rule $A \rightarrow \langle \gamma, \gamma' \rangle$.

A *derivation* under G is a sequence of translation forms:

$$\langle S_{\boxed{1}}, S_{\boxed{1}} \rangle \Rightarrow_G \langle \alpha_1, \beta_1 \rangle \Rightarrow_G \ldots \Rightarrow_G \langle \alpha_k, \beta_k \rangle$$

such that $\alpha_k \in \mathcal{T}_e^+$ and $\beta_k \in \mathcal{T}_f^+$. The string pair $\langle \alpha_k, \beta_k \rangle$ is said to be the *yield* of the derivation, and β_k is said to be a *translation* of α_k, and vice versa.

We further define the *input grammar* of G as the 4-tuple $G_e = \langle \mathcal{N}, \mathcal{T}_e, \mathcal{L}_e, S \rangle$, where $\mathcal{L}_e = \{A \to \alpha | A \to \langle \alpha, \beta \rangle \in \mathcal{L}\}$. Similarly, the *output grammar* of G is defined as the 4-tuple $G_f = \langle \mathcal{N}, \mathcal{T}_f, \mathcal{L}_f, S \rangle$, where $\mathcal{L}_f = \{A \to \beta | A \to \langle \alpha, \beta \rangle \in \mathcal{L}\}$. Both G_e and G_f are context-free grammars (CFG). We can then view synchronous parsing as a process in which two CFG parse trees are generated simultaneously, one based on the input grammar, and the other based on the output grammar. Furthermore, the two parse trees are *isomorphic*, since there is a one-to-one mapping between the non-terminal nodes in the two parse trees.

The language translation task can be formulated as follows: Given an input string x, we find a derivation under G_e that is consistent with x (if any):

$$S \Rightarrow_{G_e} \alpha_1 \Rightarrow_{G_e} \ldots \Rightarrow_{G_e} x$$

This derivation corresponds to the following derivation under G:

$$\langle S_{\boxed{1}}, S_{\boxed{1}} \rangle \Rightarrow_G \langle \alpha_1, \beta_1 \rangle \Rightarrow_G \ldots \Rightarrow_G \langle x, y \rangle$$

The string y is then a translation of x.

As a concrete example, suppose that G is the following:

$$\begin{aligned}
\mathcal{N} &= \{\text{S}, \text{NP}, \text{VP}\} \\
\mathcal{T}_e &= \{\textit{wo}, \textit{shui guo}, \textit{xi huan}\} \\
\mathcal{T}_f &= \{\textit{I}, \textit{fruits}, \textit{like}\} \\
\mathcal{L} &= \{\text{S} \to \langle\ \text{NP}_{\boxed{1}}\ \text{VP}_{\boxed{2}}\ ,\ \text{NP}_{\boxed{1}}\ \text{VP}_{\boxed{2}}\ \rangle, \\
&\quad\ \text{NP} \to \langle\ \textit{wo}\ ,\ \textit{I}\ \rangle, \\
&\quad\ \text{NP} \to \langle\ \textit{shui guo}\ ,\ \textit{fruits}\ \rangle, \\
&\quad\ \text{VP} \to \langle\ \textit{xi huan}\ \text{NP}_{\boxed{1}}\ ,\ \textit{like}\ \text{NP}_{\boxed{1}}\ \rangle\} \\
S &= \text{S}
\end{aligned}$$

Given an input string, *wo xi huan shui guo*, a derivation under G that is consistent with the input string would be:

$$\begin{aligned}
\langle\ \text{S}_{\boxed{1}}\ ,\ \text{S}_{\boxed{1}}\ \rangle &\Rightarrow_G \langle\ \text{NP}_{\boxed{1}}\ \text{VP}_{\boxed{2}}\ ,\ \text{NP}_{\boxed{1}}\ \text{VP}_{\boxed{2}}\ \rangle \\
&\Rightarrow_G \langle\ \textit{wo}\ \text{VP}_{\boxed{1}}\ ,\ \textit{I}\ \text{VP}_{\boxed{1}}\ \rangle \\
&\Rightarrow_G \langle\ \textit{wo xi huan}\ \text{NP}_{\boxed{1}}\ ,\ \textit{I like}\ \text{NP}_{\boxed{1}}\ \rangle \\
&\Rightarrow_G \langle\ \textit{wo xi huan shui guo}\ ,\ \textit{I like fruits}\ \rangle
\end{aligned}$$

Based on this derivation, a translation of *wo xi huan shui guo* would be *I like fruits*.

Synchronous grammars provide a natural way of capturing the hierarchical structures of a sentence and its translation, as well as the correspondence between their sub-parts. In Chapters 3–6, we will introduce algorithms for learning synchronous grammars such as SCFGs for both semantic parsing and tactical generation.

2.5 Statistical Machine Translation

Another area of research that is relevant to our work is machine translation, whose main goal is to translate one natural language into another. Machine translation (MT) is a particularly challenging task, because of the inherent ambiguity of natural languages on both sides. It has inspired a large body of research. In particular, the growing availability of parallel corpora, in which the same content is available in multiple languages, has stimulated interest in statistical methods for extracting linguistic knowledge from a large body of text. In this section, we review the main components of a typical statistical MT system.

Without loss of generality, we define machine translation as the task of translating a foreign sentence, $\mathbf{f}$, into an English sentence, $\mathbf{e}$. Obviously, there are many acceptable translations for a given $\mathbf{f}$. In statistical MT, every English sentence is a possible translation of $\mathbf{f}$. Each English sentence $\mathbf{e}$ is assigned a probability $\Pr(\mathbf{e}|\mathbf{f})$. The task of translating a foreign sentence, $\mathbf{f}$, is then to choose the English sentence, $\mathbf{e}^\star$, for which $\Pr(\mathbf{e}^\star|\mathbf{f})$ is the greatest. Traditionally, this task is divided into several more manageable sub-tasks, e.g.:

$$\mathbf{e}^\star = \arg\max_{\mathbf{e}} \Pr(\mathbf{e}|\mathbf{f}) = \arg\max_{\mathbf{e}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e}) \tag{2.3}$$

In this *noisy-channel* framework, the translation task is to find an English translation, $\mathbf{e}^\star$, such that (1) it is a well-formed English sentence, and (2) it explains $\mathbf{f}$ well. $\Pr(\mathbf{e})$ is traditionally called a *language model*, and $\Pr(\mathbf{f}|\mathbf{e})$ a *translation model*. The language modeling problem is essentially the same as in automatic speech recognition, where n-gram models are commonly used (Stolcke, 2002; Brants et al., 2007). On the other hand, translation models are unique to statistical MT, and will be the main focus of the following subsections.

2.5.1 Word-Based Translation Models

Brown et al. (1993b) present a series of five translation models which later became known as the *IBM Models*. These models are *word-based* because they

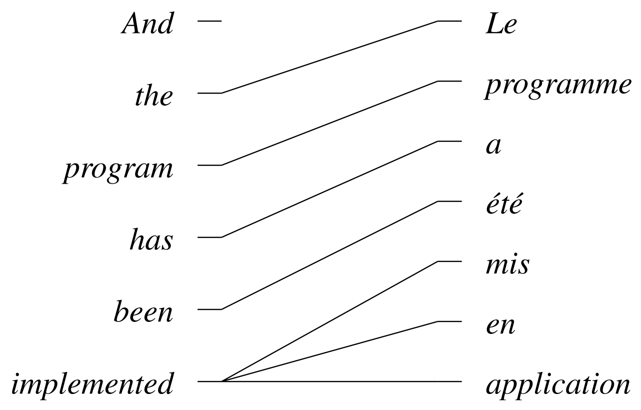


Figure 2.3: A word alignment taken from Brown et al. (1993b)

model how individual words in $\mathbf{e}$ are translated into words in $\mathbf{f}$. Such word-to-word mappings are captured in a *word alignment* (Brown et al., 1990). Suppose that $\mathbf{e} = e_1^I = \langle e_1, \ldots, e_I \rangle$, and $\mathbf{f} = f_1^J = \langle f_1, \ldots, f_J \rangle$. A word alignment, $\mathbf{a}$, between $\mathbf{e}$ and $\mathbf{f}$ is defined as:

$$\mathbf{a} = \langle a_1, \ldots, a_J \rangle \text{ where } 0 \leq a_j \leq I \text{ for all } j = 1, \ldots, J \tag{2.4}$$

where a_j is the position of the English word that the foreign word f_j is linked to. If $a_j = 0$, then f_j is not linked to any English word. Note that in the IBM Models, word alignments are constrained to be 1-to-n, i.e. each foreign word is linked to at most one English word. Figure 2.3 shows a sample word alignment for an English-French sentence pair. In this word alignment, the French word *le* is linked to the English word *the*, the French phrase *mis en application* as a whole is linked to the English word *implemented*, and so on.

The translation model $\Pr(\mathbf{f}|\mathbf{e})$ is then expressed as a sum of the probabilities of word alignments $\mathbf{a}$ between $\mathbf{e}$ and $\mathbf{f}$:

$$\Pr(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} \Pr(\mathbf{f}, \mathbf{a}|\mathbf{e}) \tag{2.5}$$

The word alignments $\mathbf{a}$ are hidden variables which must be estimated using EM. Hence $\Pr(\mathbf{f}|\mathbf{e})$ is also called a *hidden alignment model* (or *word alignment model*). The IBM Models mainly differ in terms of the formulation of $\Pr(\mathbf{f}, \mathbf{a}|\mathbf{e})$. In IBM Models 1 and 2, this probability is formulated as:

$$\Pr(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \Pr(J|\mathbf{e}) \prod_{j=1}^{J} \Pr(a_j|j, I, J) \Pr(f_j|e_{a_j}) \tag{2.6}$$

The generative process for producing $\mathbf{f}$ from $\mathbf{e}$ is as follows: Given an English sentence, $\mathbf{e}$, choose a length J for $\mathbf{f}$. Then for each foreign word position, j, choose a_j from $0, 1, \ldots, I$, and also f_j based on the English word e_{a_j}. Various simplifying assumptions are made so that inference remains tractable. In particular, a *zero-order* assumption is made such that the choice of a_j is independent of a_1^{j-1}, e.g. all word movements are independent.

The zero-order assumption of IBM Models 1 and 2 is unrealistic, as it does not take collocations into account, such as *mis en application*. In the subsequent IBM Models, this assumption is gradually relaxed, so that collocations can be better modeled. Exact inference is no longer tractable, so approximate inference must be used. Due to the complexity of these models, we will not discuss them in detail.

Word alignment models such as IBM Models 1–5 are widely used in working with parallel corpora. Among the applications are extracting parallel sentences from comparable corpora (Munteanu et al., 2004), aligning dependency-tree fragments (Ding et al., 2003), and extracting translation pairs for phrase-based and syntax-based translation models (Och and Ney, 2004; Chiang, 2005). In Chapters 3 and 4, we will show that word alignment models can be used for extracting synchronous grammar rules for semantic parsing as well.

2.5.2 Phrase-Based and Syntax-Based Translation Models

A major problem with the IBM Models is their lack of linguistic content. One approach to this problem is to introduce the concept of phrases in a *phrase-based* translation model. A basic phrase-based model translates $\mathbf{e}$ into $\mathbf{f}$ in the following steps: First, $\mathbf{e}$ is segmented into a number of sequences of consecutive words (or *phrases*), $\tilde{e}_1, \ldots, \tilde{e}_K$. These phrases are then reordered and translated into foreign phrases, $\tilde{f}_1, \ldots, \tilde{f}_K$, which are joined together to form a foreign sentence, $\mathbf{f}$. Och et al. (1999) introduce an alignment template approach in which phrase pairs, $\{\langle \tilde{e}, \tilde{f} \rangle\}$, are extracted from word alignments. The aligned phrase pairs are then generalized to form alignment templates, based on word classes learned from the training data. In Koehn et al. (2003), Tillmann (2003) and Venugopal et al. (2003), phrase pairs are extracted from word alignments without generalization. In Marcu and Wong (2002), phrase translations are learned as part of an EM algorithm in which the joint probability $\Pr(\mathbf{e}, \mathbf{f})$ is estimated.

Phrase-based translation models can be further generalized to handle hierarchical phrasal structures. Such models are collectively known as *syntax-based* translation models. Yamada and Knight (2001, 2002) present a tree-to-string translation model based on a synchronous tree-substitution grammar (Knight and Graehl, 2005). Galley et al. (2006) extends the tree-to-string model with multi-level syntactic translation rules. Chiang (2005) presents a hierarchical phrase-based model whose underlying formalism is an SCFG. Both Galley et al.'s (2006) and Chiang's (2005) systems are shown to outperform state-of-the-art phrase-based MT systems.

A common feature of syntax-based translation models is that they are all based on synchronous grammars. Synchronous grammars are ideal formalisms for formulating syntax-based translation models because they describe not only the hierarchical structures of a sentence pair, but also the correspondence between their sub-parts. In subsequent chapters, we will show that learning techniques developed for syntax-based statistical MT can be brought to bear on tasks that involve formal MRLs, such as semantic parsing and tactical generation.

Chapter 3

Semantic Parsing with Machine Translation

This chapter describes how semantic parsing can be done using statistical machine translation (Wong and Mooney, 2006). Specifically, the parsing model can be seen as a syntax-based translation model, and word alignments are used in lexical acquisition. Our algorithm is called WASP, short for *Word Alignment-based Semantic Parsing*. In this chapter, we focus on variable-free MRLs such as FUNQL and CLANG (Section 2.1). A variation of WASP that handles logical forms will be described in Chapter 4. The WASP algorithm will also form the basis of our tactical generation algorithm, WASP^{-1}, and its variants (Chapters 5 and 6).

3.1 Motivation

As mentioned in Section 2.2, prior research on semantic parsing has mainly focused on relatively simple domains such as ATIS (Section 2.1), where a typical sentence can be represented by a single semantic frame. Learning methods have been devised that can handle MRs with a complex, nested structure as in the GEOQUERY and ROBOCUP domains. However, some of these methods are based on deterministic parsing (Zelle and Mooney, 1996; Tang and Mooney, 2001; Kate et al., 2005), which lack the robustness that characterizes recent advances in statistical NLP. Other methods involve the use of fully-annotated semantically-augmented parse trees (Ge and Mooney, 2005) or prior knowledge of the NL syntax (Bos, 2005; Zettlemoyer and Collins, 2005, 2007) in training, and hence require extensive human expertise when porting to a new language or domain.

In this work, we treat semantic parsing as a language translation task. Sentences are *translated* into formal MRs through synchronous parsing (Section 2.4), which provides a natural way of capturing the hierarchical structures of NL sentences and their MRL translations, as well as the correspondence between their sub-parts. Originally developed as a theory of compilers in which syntax analysis and code generation are combined into a single phase (Aho and Ullman, 1972), synchronous parsing has seen a surge of interest recently in the machine translation community as a way of formalizing syntax-based translation models (Wu, 1997; Chiang, 2005). We argue that synchronous parsing can also be useful in translation tasks that involve *both* natural and formal languages, and in semantic parsing in particular.

In subsequent sections, we present a learning algorithm for semantic parsing called WASP. The input to the learning algorithm is a set of training sentences paired with their correct MRs. The output from the learning algorithm is a sychronous context-free grammar (SCFG), together with parameters that define a log-linear distribution over parses under the grammar. The learning algorithm assumes that an unambiguous, context-free grammar (CFG) of the target MRL is available, but it does not require any prior knowledge of the NL syntax or annotated parse trees in the training data. Experiments show that WASP performs favorably in terms of both accuracy and coverage compared to other methods requiring similar supervision, and is considerably more robust than methods based on deterministic parsing.

```
((bowner our {4}) (do our {6} (pos (left (half our)))))
```
If our player 4 has the ball, then our player 6 should stay in the left side of our half.

Figure 3.1: A meaning representation in CLANG and its English gloss

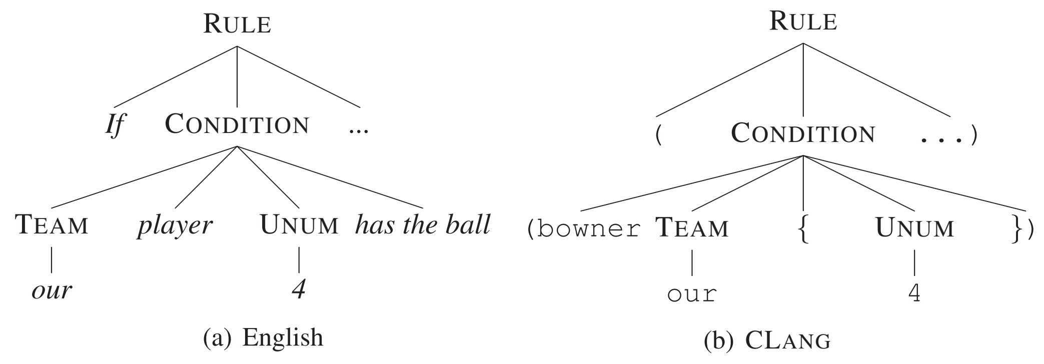


Figure 3.2: Partial parse trees for the string pair in Figure 3.1

3.2 The WASP Algorithm

To describe the WASP semantic parsing algorithm, it is best to start with an example. Consider the task of translating the English sentence in Figure 3.1 into its CLANG representation in the ROBOCUP domain. To achieve this task, we may first analyze the syntactic structure of the English sentence using a semantic grammar (Section 2.2.2) , whose non-terminals are those in the CLANG grammar. The meaning of the sentence is then obtained by combining the meanings of its subparts based on the semantic parse. Figure 3.2(a) shows a possible semantic parse of the sample sentence (the UNUM non-terminal in the parse tree stands for "uniform number"). Figure 3.2(b) shows the corresponding CLANG parse tree from which the MR is constructed.

This translation process can be formalized as synchronous parsing. A detailed description of the synchronous parsing framework can be found in Section

2.4. Under this framework, a derivation yields two strings, one for the source NL, and one for the target MRL. Given an input sentence, $\mathbf{e}$, the task of semantic parsing is to find a derivation that yields a string pair, $\langle \mathbf{e}, \mathbf{f} \rangle$, so that $\mathbf{f}$ is an MRL translation of $\mathbf{e}$. To finitely specify a potentially infinite set of string pairs, we use a weighted SCFG, G, defined by a 6-tuple:

$$G = \langle \mathcal{N}, \mathcal{T}_e, \mathcal{T}_f, \mathcal{L}, S, \lambda \rangle \tag{3.1}$$

where $\mathcal{N}$ is a finite set of non-terminal symbols, $\mathcal{T}_e$ is a finite set of NL terminal symbols (words), $\mathcal{T}_f$ is a finite set of MRL terminal symbols, $\mathcal{L}$ is a lexicon which consists of a finite set of rules[1], $S \in \mathcal{N}$ is a distinguished start symbol, and λ is a set of parameters that define a probability distribution over derivations under G. Each rule in $\mathcal{L}$ takes the following form:

$$A \rightarrow \langle \alpha, \beta \rangle \tag{3.2}$$

where $A \in \mathcal{N}$, $\alpha \in (\mathcal{N} \cup \mathcal{T}_e)^+$, and $\beta \in (\mathcal{N} \cup \mathcal{T}_f)^+$. The LHS of the rule is a non-terminal, A. The RHS of the rule is a pair of strings, $\langle \alpha, \beta \rangle$, in which the non-terminals in α are a permutation of the non-terminals in β. Below are some SCFG rules that can be used to produce the parse trees in Figure 3.2:

$$\begin{aligned}
\textsc{Rule} &\rightarrow \langle\ \textit{if}\ \textsc{Condition}_{\boxed{1}}\ ,\ \textsc{Directive}_{\boxed{2}}\ .\ , \\
&\qquad \texttt{(}\textsc{Condition}_{\boxed{1}}\ \textsc{Directive}_{\boxed{2}}\texttt{)}\ \rangle \\
\textsc{Condition} &\rightarrow \langle\ \textsc{Team}_{\boxed{1}}\ \textit{player}\ \textsc{Unum}_{\boxed{2}}\ \textit{has}\ (1)\ \textit{ball}\ , \\
&\qquad \texttt{(bowner}\ \textsc{Team}_{\boxed{1}}\ \texttt{\{}\textsc{Unum}_{\boxed{2}}\texttt{\})}\ \rangle \\
\textsc{Team} &\rightarrow \langle\ \textit{our}\ ,\ \texttt{our}\ \rangle \\
\textsc{Unum} &\rightarrow \langle\ \textit{4}\ ,\ \texttt{4}\ \rangle
\end{aligned}$$

[1]Henceforth, we reserve the term *rules* for production rules of an SCFG, and the term *productions* for production rules of an ordinary CFG.

Each SCFG rule $A \rightarrow \langle \alpha, \beta \rangle$ is a combination of a production of the NL semantic grammar, $A \rightarrow \alpha$, and a production of the MRL grammar, $A \rightarrow \beta$. We call the string α an *NL string*, and the string β an *MR string*. Non-terminals in NL and MR strings are indexed with $\boxed{1}, \boxed{2}, \ldots$ to show their association. All derivations start with a pair of associated start symbols, $\langle S_{\boxed{1}}, S_{\boxed{1}} \rangle$. Each step of a derivation involves the rewriting of a pair of associated non-terminals. Below is a derivation that yields the sample English sentence and its CLANG representation in Figure 3.1:

$\langle$ RULE$_{\boxed{1}}$, RULE$_{\boxed{1}}$ $\rangle$
$\Rightarrow \langle$ *if* CONDITION$_{\boxed{1}}$, DIRECTIVE$_{\boxed{2}}$. ,
`(`CONDITION$_{\boxed{1}}$ DIRECTIVE$_{\boxed{2}}$`)` $\rangle$
$\Rightarrow \langle$ *if* TEAM$_{\boxed{1}}$ *player* UNUM$_{\boxed{2}}$ *has the ball* , DIRECTIVE$_{\boxed{3}}$. ,
`((bowner` TEAM$_{\boxed{1}}$ `{`UNUM$_{\boxed{2}}$`})` DIRECTIVE$_{\boxed{3}}$`)` $\rangle$
$\Rightarrow \langle$ *if our player* UNUM$_{\boxed{1}}$ *has the ball* , DIRECTIVE$_{\boxed{2}}$. ,
`((bowner our {`UNUM$_{\boxed{1}}$`})` DIRECTIVE$_{\boxed{2}}$`)` $\rangle$
$\Rightarrow \langle$ *if our player 4 has the ball* , DIRECTIVE$_{\boxed{1}}$. ,
`((bowner our {4})` DIRECTIVE$_{\boxed{1}}$`)` $\rangle$
$\Rightarrow$ ...
$\Rightarrow \langle$ *if our player 4 has the ball, then our player 6 should stay in the left side of our half.* ,
```
((bowner our {4})
 (do our {6} (pos (left (half our))))) 
```
$\rangle$

Here the CLANG representation is said to be a *translation* of the English sentence. Given an NL sentence, $\mathbf{e}$, there can be multiple derivations that yield $\mathbf{e}$ (and thus multiple MRL translations of $\mathbf{e}$). To discriminate the correct translation from the incorrect ones, we use a probabilistic model, parameterized by λ, that takes a derivation, $\mathbf{d}$, and returns its likelihood of being correct. The output translation, $\mathbf{f}^{\star}$, of a sentence, $\mathbf{e}$, is defined as:

$$\mathbf{f}^\star = \mathbf{f}\left(\operatorname*{arg\,max}_{\mathbf{d}\in D(G|\mathbf{e})} \Pr_\lambda(\mathbf{d}|\mathbf{e})\right) \tag{3.3}$$

where $\mathbf{f}(\mathbf{d})$ is the MR string that a derivation $\mathbf{d}$ yields, and $D(G|\mathbf{e})$ is the set of all derivations of G that yield $\mathbf{e}$. In other words, the output MRL translation is the yield of the most probable derivation that yields the input NL sentence. This formulation is chosen because $\mathbf{f}^\star$ can be efficiently computed using a dynamic-programming algorithm (Viterbi, 1967).

Since $\mathcal{N}$, $\mathcal{T}_e$, $\mathcal{T}_f$ and S are fixed given an NL and an MRL, we only need to learn a lexicon, $\mathcal{L}$, and a probabilistic model parameterized by λ. A lexicon defines the set of derivations that are possible, so the induction of a probabilistic model requires a lexicon in the first place. Therefore, the learning task can be divided into the following two sub-tasks:

1. Acquire a lexicon, $\mathcal{L}$, which implicitly defines the set of all possible derivations, $D(G)$.
2. Learn a set of parameters, λ, that define a probability distribution over derivations in $D(G)$.

Both sub-tasks require a training set, $\{\langle \mathbf{e}_i, \mathbf{f}_i\rangle\}$, where each training example $\langle \mathbf{e}_i, \mathbf{f}_i\rangle$ is an NL sentence, $\mathbf{e}_i$, paired with its correct MR, $\mathbf{f}_i$. Lexical acquisition also requires an unambiguous CFG of the MRL. Since there is no lexicon to begin with, it is not possible to include correct derivations in the training data. Therefore, these derivations are treated as hidden variables which must be estimated through EM-type iterative training, and the learning task is not fully supervised. Figure 3.3 gives an overview of the WASP semantic parsing algorithm.

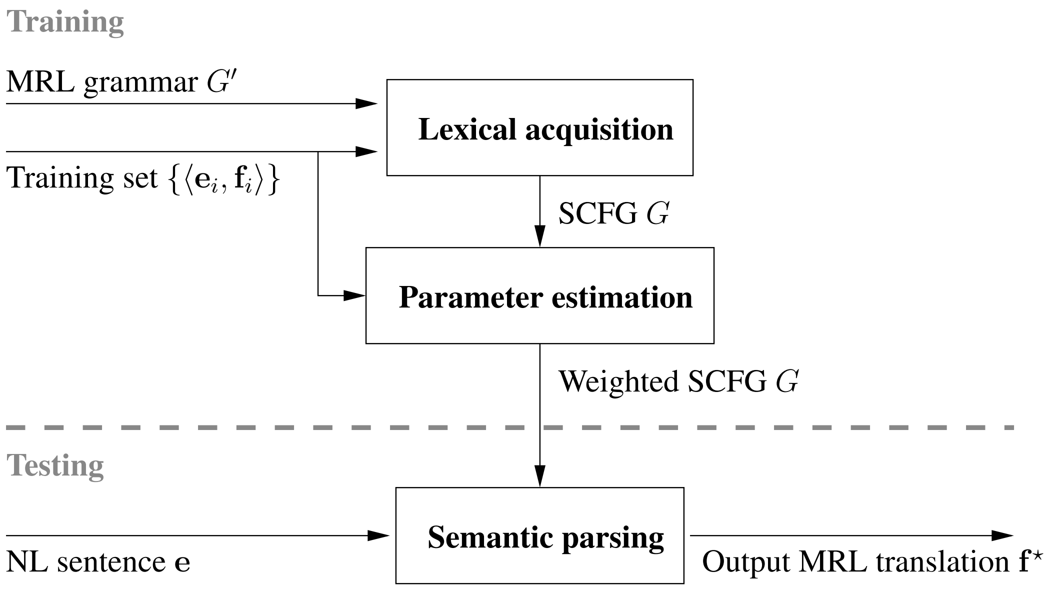


Figure 3.3: Overview of the WASP semantic parsing algorithm

In Sections 3.2.1–3.2.3, we will focus on lexical acquisition. We will describe the probabilistic model in Section 3.2.4.

3.2.1 Lexical Acquisition

A lexicon is a mapping from words to their meanings. In Section 2.5.1, we showed that word alignments can be used for defining a mapping from words to their meanings. In WASP, we use word alignments for lexical acquisition. The basic idea is to train a statistical word alignment model on the training set, and then find the most probable word alignments for each training example. A lexicon is formed by extracting SCFG rules from these word alignments (Chiang, 2005).

Let us illustrate this algorithm using an example. Suppose that we are given the string pair in Figure 3.1 as the training data. The word alignment model is to find a word alignment for this string pair. A sample word alignment is shown in Figure 3.4, where each CLang symbol is treated as a word. This presents three difficulties. First, not all MR symbols carry specific meanings. For example, in CLang, parentheses (`(`, `)`) and braces (`{`, `}`) are delimiters that are semantically vacuous. Such symbols are not supposed to be aligned with any words, and inclusion of these symbols in the training data is likely to confuse the word alignment model. Second, not all concepts have an associated MR symbol. For example, in CLang, the mere appearance of a condition followed by a directive indicates an *if-then* rule, and there is no CLang predicate associated with the concept of an if-then rule. Third, multiple concepts may be associated with the same MR symbol. For example, the CLang predicate `pt` is polysemous. Its meaning depends on the types of arguments it is given. It specifies the xy-coordinates when its arguments are two numbers (e.g. `(pt 0 0)`), the current position of the ball when its argument is the MR symbol `ball` (i.e. `(pt ball)`), or the current position of a player when a team and a uniform number are given as arguments (e.g. `(pt our 4)`). Judging from the `pt` symbol alone, the word alignment model would not be able to identify its exact meaning.

A simple, principled way to avoid these difficulties is to represent an MR using a sequence of MRL productions used to generate it. This sequence corresponds to the top-down, left-most derivation of an MR. Each MRL production is then treated as a word. Figure 3.5 shows a word alignment between the sample sentence and the linearized parse of its CLang representation. Here the second production, Condition → `(bowner` Team `{`Unum`})`, is the one that rewrites the Condition non-terminal in the first production, Rule → `(`Condition Directive`)`, and so on. Treating MRL productions as words allows collocations to be treated as a single lexical unit (e.g. the symbols `(`, `pt`, `ball`, followed by

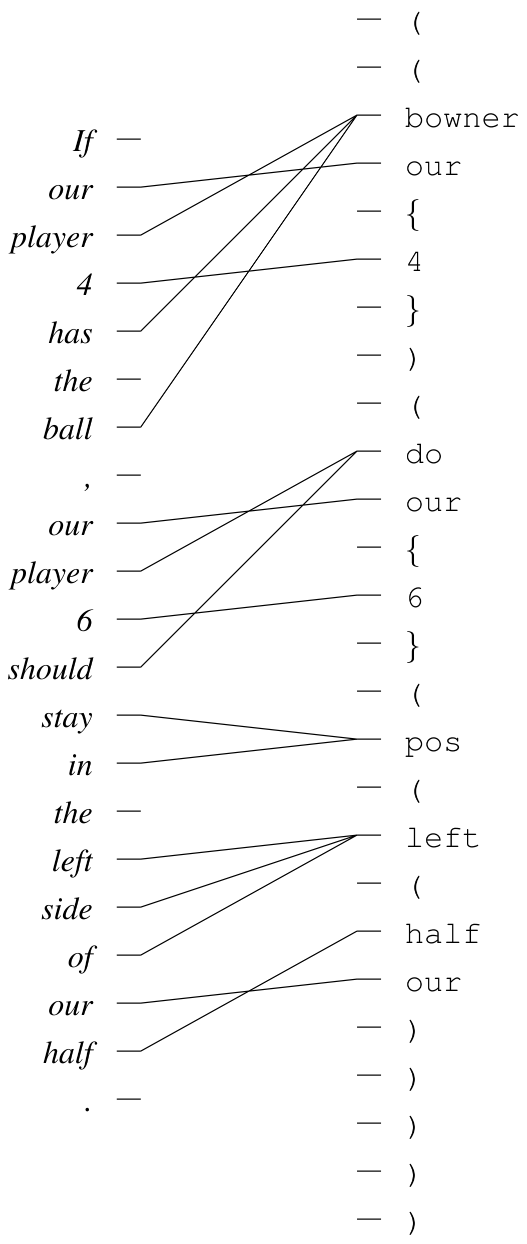


Figure 3.4: A word alignment between English words and CLANG symbols

`))`. A lexical unit can be discontiguous (e.g. `(`, `pos`, followed by a region, and then the symbol `))`). It also allows the meaning of a polysemous MR symbol to be disambiguated, where each possible meaning corresponds to a distinct MRL production. In addition, it allows productions that are unlexicalized (e.g. RULE $\rightarrow$ `(`CONDITION DIRECTIVE`))` to be associated with some English words. Note that for each MR there is a unique parse tree, since the MRL grammar is unambiguous. Also note that the structure of a MR parse tree is preserved through linearization. The structural aspect of an MR parse tree will play an important role in the subsequent extraction of SCFG rules.

Word alignments can be obtained using any off-the-shelf word alignment model. In this work, we use the GIZA++ implementation (Och and Ney, 2003) of IBM Model 5 (Brown et al., 1993b).

Assuming that each NL word is linked to at most one MRL production, SCFG rules are extracted from a word alignment in a bottom-up manner. The process starts with productions with no non-terminals on the RHS, e.g. TEAM $\rightarrow$ `our` and UNUM $\rightarrow$ `4`. For each of these productions, $A \rightarrow \beta$, an SCFG rule $A \rightarrow \langle \alpha, \beta \rangle$ is extracted such that α consists of the words to which the production is linked. For example, the following rules would be extracted from Figure 3.5:

TEAM $\rightarrow \langle$ *our* , `our` $\rangle$
UNUM $\rightarrow \langle$ *4* , `4` $\rangle$
UNUM $\rightarrow \langle$ *6* , `6` $\rangle$

Next we consider productions with non-terminals on the RHS, i.e. predicates with arguments. In this case, the NL string α consists of the words to which the production is linked, as well as non-terminals showing where the arguments are realized. For example, for the `bowner` predicate, the extracted rule would be:

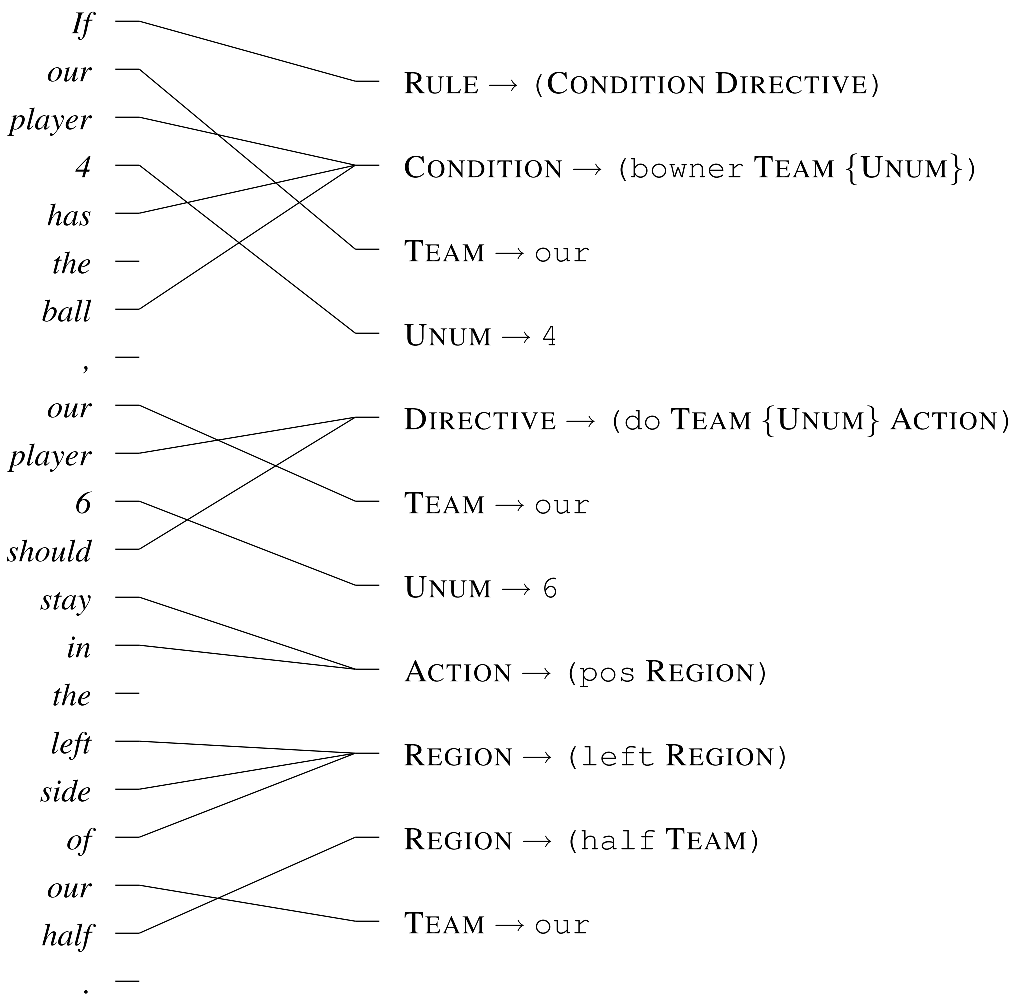


Figure 3.5: A word alignment between English words and CLANG productions

$$\textsc{Condition} \rightarrow \langle\ \textsc{Team}_{\boxed{1}}\ \textit{player}\ \textsc{Unum}_{\boxed{2}}\ \textit{has}\ (1)\ \textit{ball}\ ,\ \texttt{(bowner}\ \textsc{Team}_{\boxed{1}}\ \texttt{\{}\textsc{Unum}_{\boxed{2}}\texttt{\})}\ \rangle$$

where (1) denotes a *word gap* of size 1, due to the unaligned word *the* that comes between *has* and *ball*. Formally, a word gap of size g can be seen as a special non-terminal that expands to at most g NL words, which allows for some flexibility during pattern matching. Note the use of indices to indicate the association between non-terminals in the extracted NL and MR strings.

Similarly, the following SCFG rules would be extracted from the same word alignment:

$$\begin{aligned}
\textsc{Region} &\rightarrow \langle\ \textsc{Team}_{\boxed{1}}\ \textit{half}\ ,\ \texttt{(half}\ \textsc{Team}_{\boxed{1}}\texttt{)}\ \rangle \\
\textsc{Region} &\rightarrow \langle\ \textit{left side of}\ \textsc{Region}_{\boxed{1}}\ ,\ \texttt{(left}\ \textsc{Region}_{\boxed{1}}\texttt{)}\ \rangle \\
\textsc{Action} &\rightarrow \langle\ \textit{stay in}\ (1)\ \textsc{Region}_{\boxed{1}}\ ,\ \texttt{(pos}\ \textsc{Region}_{\boxed{1}}\texttt{)}\ \rangle \\
\textsc{Directive} &\rightarrow \langle\ \textsc{Team}_{\boxed{1}}\ \textit{player}\ \textsc{Unum}_{\boxed{2}}\ \textit{should}\ \textsc{Action}_{\boxed{3}}\ , \\
&\qquad \texttt{(do}\ \textsc{Team}_{\boxed{1}}\ \texttt{\{}\textsc{Unum}_{\boxed{2}}\texttt{\}}\ \textsc{Action}_{\boxed{3}}\texttt{)}\ \rangle \\
\textsc{Rule} &\rightarrow \langle\ \textit{if}\ \textsc{Condition}_{\boxed{1}}\ (1)\ \textsc{Directive}_{\boxed{2}}\ (1)\ , \\
&\qquad \texttt{(}\textsc{Condition}_{\boxed{1}}\ \textsc{Directive}_{\boxed{2}}\texttt{)}\ \rangle
\end{aligned}$$

Note the word gap (1) at the end of the NL string in the last rule, which is due to the unaligned period in the sentence. This word gap is added because all words in a sentence have to be consumed by a derivation.

Figure 3.6 shows the basic lexical acquisition algorithm of WASP. The training set, $T = \{\langle \mathbf{e}_i, \mathbf{f}_i\rangle\}$, is used to train the alignment model M, which is in turn used to obtain the k-best word alignments for each training example (we use $k = 10$). SCFG rules are extracted from each of these word alignments. It is done in a bottom-up fashion, such that an MR predicate is processed only after its arguments have all been processed. This order is enforced by the backward traversal of a linearized MR parse. The lexicon, $\mathfrak{L}$ then consists of all rules extracted from all k-best word alignments for all training examples.

Input: a training set, $T = \{\langle \mathbf{e}_i, \mathbf{f}_i \rangle\}$, and an unambiguous MRL grammar, G'.

```
ACQUIRE-LEXICON$(T, G')$
1  $\mathfrak{L} \leftarrow \emptyset$
2  for $i \leftarrow 1$ to $|T|$
3        do $\mathbf{f}'_i \leftarrow$ linearized parse of $\mathbf{f}_i$ under $G'$
4  Train a word alignment model, $M$, using $\{\langle \mathbf{e}_i, \mathbf{f}'_i \rangle\}$ as the training set
5  for $i \leftarrow 1$ to $|T|$
6        do $\mathbf{a}^\star_{1,\ldots,k} \leftarrow k$-best word alignments for $\langle \mathbf{e}_i, \mathbf{f}'_i \rangle$ under $M$
7           for $k' \leftarrow 1$ to $k$
8               do for $j \leftarrow |\mathbf{f}'_i|$ downto $1$
9                      do $A \leftarrow \mathrm{lhs}(f'_{ij})$
10                        $\alpha \leftarrow$ words to which $f'_{ij}$ and its arguments are linked in $\mathbf{a}^\star_{k'}$
11                        $\beta \leftarrow \mathrm{rhs}(f'_{ij})$
12                        $\mathfrak{L} \leftarrow \mathfrak{L} \cup \{A \rightarrow \langle \alpha, \beta \rangle\}$
13                        Replace $\alpha$ with $A$ in $\mathbf{a}^\star_{k'}$
14 return $\mathfrak{L}$
```

Figure 3.6: The basic lexical acquisition algorithm of WASP

3.2.2 Maintaining Parse Tree Isomorphism

There are two cases where the ACQUIRE-LEXICON procedure would not extract any rules for a production p:

1. None of the descendants of p in the MR parse tree are linked to any words.

2. The NL string associated with p covers a word w linked to a production p' that is *not* a descendant of p in the MR parse tree. Rule extraction is forbidden in this case because it would destroy the link between w and p'.

The first case arises when a concept is not realized in NL. For example, the concept of “our team” is often assumed, because advice is given from the perspective of a team coach. When we say *the goalie should always stay in our goal area*, we mean

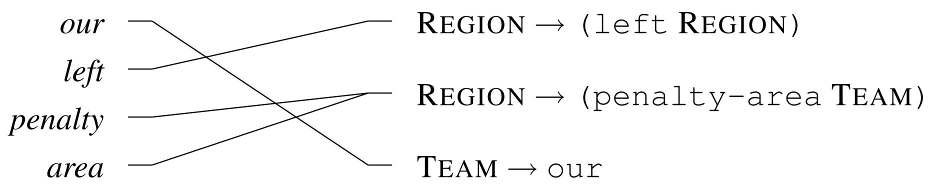


Figure 3.7: A case where the ACQUIRE-LEXICON procedure fails

our (`our`) goalie, not the other team's (`opp`) goalie. Hence the concept of `our` is often not realized. The second case arises when the NL and MR parse trees are not isomorphic. Consider the word alignment between *our left penalty area* and its CLANG representation in Figure 3.7. The extraction of the rule REGION $\rightarrow \langle$ TEAM$_{\boxed{1}}$ (1) *penalty area* , `(penalty-area` TEAM$_{\boxed{1}}$`)` $\rangle$ would destroy the link between *left* and REGION → `(left` REGION`)`. A possible explanation for this is that, syntactically, *our* modifies *left penalty area* (consider the coordination phrase *our left penalty area and right goal area*, where *our* modifies both *left penalty area* and *right goal area*). But conceptually, "left" modifies the concept of "our penalty area" by referring to its left half. Note that the NL and MR parse trees must be isomorphic under the SCFG formalism (Section 2.4.1).

The NL and MR parse trees can be made isomorphic by merging nodes in the MR parse tree, combining several productions into one. For example, since no rules can be extracted for the production REGION → `(penalty-area` TEAM`)`, it is combined with its parent node to form REGION → `(left (penalty-area` TEAM`))`, for which an NL string TEAM *left penalty area* is extracted. In general, the merging process continues until a rule is extracted from the merged node. Assuming the alignment is not empty, the process is guaranteed to end with a rule extracted.

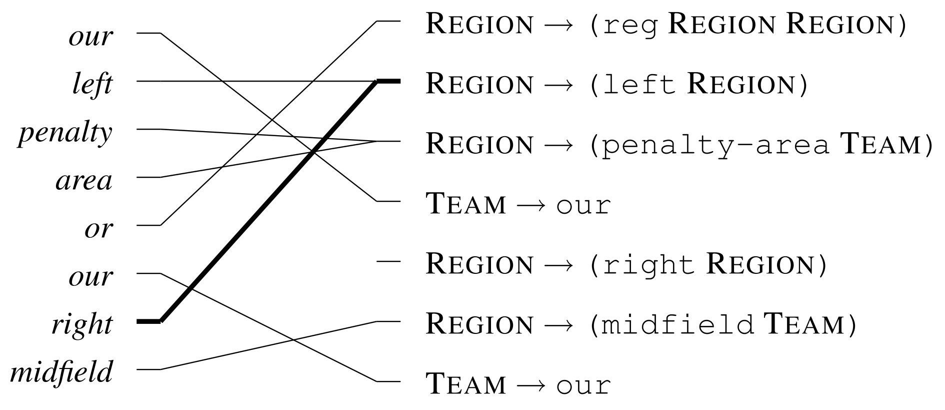


Figure 3.8: A case where a bad link disrupts phrasal coherence

3.2.3 Phrasal Coherence

The effectiveness of the lexical acquisition algorithm described so far critically depends on whether the word alignment model observes phrasal coherence. This means words that are linked to a predicate and its arguments should stay close together. Moreover, these words should form a hierarchical phrase structure that is roughly isomorphic to the MR parse tree. Any major disruption of phrasal coherence would lead to excessive node merging (Section 3.2.2), which is a major cause of overfitting. For example, in Figure 3.8, the word *right* is far from *left penalty area*, yet it is linked to the `left` predicate (shown as a thick line). This link crosses many other links in the word alignment, forcing many nodes in the MR parse tree to merge (e.g. `left` with `reg`, `midfield` with `right` and then with `reg`). The resulting SCFG rule, REGION → $\langle$ TEAM$_{\boxed{1}}$ *left penalty area or* TEAM$_{\boxed{2}}$ *right midfield* , `(reg (left (penalty-area` TEAM$_{\boxed{1}}$`)) (right (midfield` TEAM$_{\boxed{2}}$`)))` $\rangle$, is very long and does not generalize well to other cases of region union (`reg`).

Ideally, this problem can be solved using a word alignment model that strictly observes phrasal coherence. However, this often requires rules that model the reordering of tree nodes (i.e. synchronous grammars), which are exactly what WASP is trying to learn. Our goal is to bootstrap the learning process by using a simpler, word-based alignment model that produces a generally coherent alignment, and then remove links that could cause excessive node merging. This is done before rule extraction takes place.

The link removal algorithm works as follows. Recall that rule extraction from a word alignment, $\mathbf{a}$, is forbidden where the NL string associated with a production, p, covers a word linked to a production that is not a descendant of p in the MR parse tree. We call such a word a *violation* of the isomorphism constraint. For each production p in the MR parse tree, we count the number of violations that would prevent a rule from being extracted for p. Then the total number of violations for all productions in the MR parse tree is obtained, denoted by $v(\mathbf{a})$. A simple greedy procedure for removing bad links is to repeatedly remove the link $a \in \mathbf{a}$ that maximizes $v(\mathbf{a}) - v(\mathbf{a} \setminus \{a\}) > 0$, until $v(\mathbf{a})$ cannot be further reduced. A link stronger than a certain threshold (0.9) is never removed, so that merging of productions as in Figure 3.7 is still possible. The strength of a link $w \leftrightarrow p$ is defined as the translation probability, $\Pr(p|w)$, given by GIZA++, which is found to be highly correlated with the validity of a link. To replenish the removed links, links from a reverse alignment, $\tilde{\mathbf{a}}$ (obtained by treating the source language as target, and vice versa), are added to $\mathbf{a}$, as long as $\mathbf{a}$ remains n-to-1, and $v(\mathbf{a})$ is not increased.

The complete lexical acquisition algorithm is thus the following: Train a word alignment model, M, and a reverse word alignment model, $\tilde{M}$, using the training set, T. Obtain the k-best alignments, $\mathbf{a}^\star_{1,\ldots,k}$, and the best reverse alignment, $\tilde{\mathbf{a}}^\star$, for each training example in T using M and $\tilde{M}$. Remove bad links from each $\mathbf{a}^\star_{k'}$ and replenish the removed links by adding links from $\tilde{\mathbf{a}}^\star$. Then extract rules from $\mathbf{a}^{\star}_{1,\ldots,k}$ as described in the ACQUIRE-LEXICON procedure (lines 7–13), while merging nodes in the MR parse tree if necessary.

3.2.4 Probabilistic Model

Once a lexicon is acquired, the next task is to learn a probabilistic model for parse disambiguation. We propose a log-linear model that defines a conditional probability distribution over derivations given an input NL sentence. There has been much work on using log-linear models for NLP tasks such as part-of-speech tagging (Ratnaparkhi, 1996), syntactic parsing (Charniak, 2000; Clark and Curran, 2003), named entity recognition (Chieu and Ng, 2003), and machine translation (Koehn et al., 2003). A primary advantage of log-linear models is their flexibility. Features may interact with each other, allowing easy experimentation with different feature sets. Similar to Riezler et al. (2002), we will train our log-linear model on incomplete data, since derivations are not observed in the training data. It is the *yields* of these derivations—NL sentences and their MRs—that we observe.

In our log-linear model, the conditional probability of a derivation, $\mathbf{d}$, given an input sentence, $\mathbf{e}$, is defined as:

$$\Pr_{\lambda}(\mathbf{d}|\mathbf{e}) = \frac{1}{Z_{\lambda}(\mathbf{e})} \exp \sum_i \lambda_i f_i(\mathbf{d}) \tag{3.4}$$

where f_i is a *feature function* (or *feature* for short) that returns a real value given a derivation, and $Z_{\lambda}(\mathbf{e})$ is a normalizing factor such that the conditional probabilities sum to one over all derivations that yield $\mathbf{e}$. We use the following feature types:

- For each rule $r \in \mathcal{L}$, there is a feature, f_r, that returns the number of times r is used in a derivation.
- For each word $w \in \mathcal{T}_e$, there is a feature, f_w, that returns the number of times w is generated from word gaps in a derivation.

- Generation of words not previously encountered during training is modeled using an extra feature, f_*, that returns the *total* number of words generated from word gaps in a derivation.

In WASP, since the output grammar of a learned SCFG is the target MRL grammar, all MRL translations are well-formed to begin with. So the probabilistic model can be relatively simple. The number of features that we use in our log-linear model is quite modest (less than 3,000 in our experiments). A similar set of features is also used by Zettlemoyer and Collins (2005).

The output MRL translation, $\mathbf{f}^\star$, given a sentence, $\mathbf{e}$, is the yield of the most probable derivation that yields $\mathbf{e}$ (cf. Equation 3.3):

$$\begin{aligned}\mathbf{f}^\star &= \mathbf{f}\left(\operatorname*{arg\,max}_{\mathbf{d}\in D(G|\mathbf{e})} \exp \sum_i \lambda_i f_i(\mathbf{d})\right) \\ &= \mathbf{f}\left(\operatorname*{arg\,max}_{\mathbf{d}\in D(G|\mathbf{e})} \sum_i \lambda_i f_i(\mathbf{d})\right) \qquad (3.5)\end{aligned}$$

where $D(G|\mathbf{e})$ is the set of derivations under G that yield $\mathbf{e}$. The output translation can be easily computed using the Viterbi algorithm (Viterbi, 1967), with an Earley chart (Earley, 1970; Stolcke, 1995) that keeps track of derivations that are consistent with the input string. Decoding takes cubic time with respect to the sentence length.

The model parameters, λ, are estimated by maximizing the conditional log-likelihood of the training set (Berger et al., 1996; Riezler et al., 2002):

$$\lambda^\star = \operatorname*{arg\,max}_{\lambda} \sum_{\langle \mathbf{e}_j, \mathbf{f}_j\rangle \in T} \log \Pr_\lambda(\mathbf{f}_j|\mathbf{e}_j) \qquad (3.6)$$

Expanding the conditional log-likelihood, we get:

$$\begin{aligned}
L(\lambda) &= \sum_{\langle \mathbf{e}_j, \mathbf{f}_j \rangle \in T} \log \Pr_\lambda(\mathbf{f}_j | \mathbf{e}_j) \\
&= \sum_{\langle \mathbf{e}_j, \mathbf{f}_j \rangle \in T} \log \sum_{\mathbf{d} \in D(G|\mathbf{e}_j, \mathbf{f}_j)} \Pr_\lambda(\mathbf{d} | \mathbf{e}_j) \\
&= \sum_{\langle \mathbf{e}_j, \mathbf{f}_j \rangle \in T} \left(\log \left(\sum_{\mathbf{d} \in D(G|\mathbf{e}_j, \mathbf{f}_j)} \exp \sum_i \lambda_i f_i(\mathbf{d}) \right) - \log Z_\lambda(\mathbf{e}_j) \right) \\
&= \sum_{\langle \mathbf{e}_j, \mathbf{f}_j \rangle \in T} \left(\log \left(\sum_{\mathbf{d} \in D(G|\mathbf{e}_j, \mathbf{f}_j)} \exp \sum_i \lambda_i f_i(\mathbf{d}) \right) \right. \\
&\qquad \left. - \log \left(\sum_{\mathbf{d} \in D(G|\mathbf{e}_j)} \exp \sum_i \lambda_i f_i(\mathbf{d}) \right) \right)
\end{aligned}$$

where $D(G|\mathbf{e}, \mathbf{f})$ is the set of derivations under G that yield $\langle \mathbf{e}, \mathbf{f} \rangle$ (hence $D(G|\mathbf{e}, \mathbf{f}) \subseteq D(G|\mathbf{e})$). Differentiating L with respect to λ_i gives:

$$\frac{\partial}{\partial \lambda_i} L(\lambda) = \sum_{\langle \mathbf{e}_j, \mathbf{f}_j \rangle \in T} \left(\sum_{\mathbf{d} \in D(G|\mathbf{e}_j, \mathbf{f}_j)} \Pr_\lambda(\mathbf{d} | \mathbf{e}_j, \mathbf{f}_j) f_i(\mathbf{d}) - \sum_{\mathbf{d} \in D(G|\mathbf{e}_j)} \Pr_\lambda(\mathbf{d} | \mathbf{e}_j) f_i(\mathbf{d}) \right)$$

which is the difference between the expectations of $f_i(\mathbf{d})$ with respect to the distributions $\Pr_\lambda(\mathbf{d} | \mathbf{e}_j, \mathbf{f}_j)$ and $\Pr_\lambda(\mathbf{d} | \mathbf{e}_j)$. Locally optimal parameters $\lambda^\star$ can then be found by using gradient-based methods such as gradient ascent, conjugate gradient, and quasi-Newton methods. In our experiments, we use the L-BFGS algorithm (Nocedal, 1980) to compute $\lambda^\star$. L-BFGS is a limited-memory quasi-Newton method which implicitly approximates the Hessian matrix based on previous values of L and L'. It has shown good convergence properties in various NLP-related optimization tasks (Malouf, 2002).

Computation of L and L' requires statistics that depend on $D(G|\mathbf{e}_j, \mathbf{f}_j)$ and $D(G|\mathbf{e}_j)$. Since both sets can be extremely large, it is not feasible to enumerate them. However, using a similar parsing chart used for decoding, it is possible to obtain the required statistics using dynamic-programming techniques similar to the Inside-Outside algorithm (Miyao and Tsujii, 2002). In particular, computation that involves $D(G|\mathbf{e}_j, \mathbf{f}_j)$ can be done by keeping track of MR translations inside chart items, and allowing chart items to combine only when it results in a substring of $\mathbf{f}_j$.

A Gaussian prior ($\sigma^2 = 100$) is used to regularize the log-linear model (Chen and Rosenfeld, 1999). Unlike the fully-supervised case, the conditional log-likelihood L is not concave with respect to λ, so the optimization algorithm is sensitive to initial parameters. To assume as little as possible, λ is initialized to $\mathbf{0}$. Following Zettlemoyer and Collins (2005), only rules that are used in the most probable derivations for each training example are retained in the final lexicon. All other rules are discarded. This heuristic is used to improve accuracy of the semantic parser, assuming that rules used in the most probable derivations are the most accurate.

In summary, the WASP learning algorithm is divided into two sub-tasks. The first sub-task is to acquire a lexicon consisting of SCFG rules extracted from word alignments between training sentences and their correct MRs. The second sub-task is to estimate the parameters that define a log-linear distribution over parses under the learned SCFG. The resulting weighted SCFG can then be used for parsing novel sentences.

3.3 Experiments

This section describes the experiments that were performed to demonstrate the effectiveness of the WASP semantic parsing algorithm.

3.3.1 Data Sets

We evaluated WASP in the GEOQUERY and ROBOCUP domains (Section 2.1). The GEOQUERY corpus consists of 880 English questions gathered from various sources. 250 of them were gathered from an undergraduate language class (Zelle and Mooney, 1996). These questions were manually translated into a logical query language based on Prolog (Appendix A.1). An additional 630 English questions were subsequently gathered from an undergraduate AI class, and from users of a web interface to a CHILL (Zelle and Mooney, 1996) prototype trained on the initial 250 data set (Tang and Mooney, 2001). These questions together with their Prolog logical forms and the original 250 data set, form a larger 880-example data set. Queries in the 250 data set were also translated into Spanish and Turkish, each by a native speaker of the language, and into Japanese by an English native speaker who had learned Japanese as a second language.

Since WASP can only handle variable-free MRLs, we wrote a small program that translates Prolog logical forms into a functional query language (FUNQL) developed for the GEOQUERY domain (Appendix A.2).

For the ROBOCUP domain, the corpus consists of 300 pieces of coaching advice encoded in CLANG (Appendix A.3), randomly selected from the log files of the 2003 ROBOCUP Coach Competition. Each formal statement was manually translated into English by one of four annotators (Kate et al., 2005). Basically, CLANG statements are variable-free. The CLANG language does allow the use of logical variables, but they have very limited use and rarely occur in the data.

Table 3.1 shows some statistics of the corpora used for evaluating WASP. Note that sentences in the ROBOCUP data set are much longer than those in GEOQUERY on average.

	GEOQUERY					ROBOCUP
MRL	FUNQL					CLANG
# non-terminals	13					12
# productions	133					134
# examples	250				880	300
NL	English	Spanish	Japanese	Turkish	English	English
Avg. sent. length	6.87	7.39	9.11	5.76	7.57	22.52
# unique words	165	159	158	220	280	337

Table 3.1: Corpora used for evaluating WASP

3.3.2 Methodology

We performed standard 10-fold cross validation in our experiments. During testing, we counted the number of sentences for which there was an output MRL translation. Translation fails when there are constructs in a sentence that a learned parser does not cover. We also counted the number of output MRL translations that were correct. For GEOQUERY, a translation is correct if it retrieves the same answer from the GEOQUERY database as the reference query. For ROBOCUP, a translation is correct if it exactly matches the correct MR, up to reordering of arguments for commutative predicates like `and`. These strict criteria were chosen because two slightly different representations can have very different meanings (e.g. negation). Based on these counts, we compute the *precision*, *recall* and *F-measure* of a learned parser:

$$\text{Precision} = \frac{\text{No. of correct output translations}}{\text{No. of output translations}} \tag{3.7}$$

$$\text{Recall} = \frac{\text{No. of correct output translations}}{\text{No. of test sentences}} \tag{3.8}$$

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.9}$$

For each domain, there is a minimal set of *initial rules* representing knowledge needed for translating basic domain entities. These rules are always included in a lexicon. For GEOQUERY, the initial rules are the following:

CITYNAME → $\langle \mathbf{e}(c), c \rangle$, for all city names c
e.g. for English: ⟨ *new york* , `'new york'` ⟩
for Japanese: ⟨ *nyuu yooku* , `'new york'` ⟩
RIVERNAME → $\langle \mathbf{e}(r), r \rangle$, for all river names r
STATENAME → $\langle \mathbf{e}(s), s \rangle$, for all state names s

Similar rules for lake names, mountain names, state name abbreviations, and other place names.

Here $\mathbf{e}(x)$ is an NL expression that corresponds to x. Since the GEOQUERY database is in English, $\mathbf{e}(x) = x$ for English. For other languages, $\mathbf{e}(x)$ can be different. For example, $\mathbf{e}($`'new york'`$)$ is *nyuu yooku* in Japanese. A rule such as CITYNAME → ⟨ *nyuu yooku* , `'new york'` ⟩ provides domain knowledge that cannot be easily learned without analyzing the phonological features of a name. Such initial rules can be easily constructed from a bilingual dictionary. Note that a name can be ambiguous. For example, *New York* can be either a state or a city. A semantic parser needs to disambiguate between these two cases based on surrounding context.

For ROBOCUP, the initial rules are the following:

UNUM → $\langle i, i \rangle$, for all integers $i = 1, \ldots, 11$
NUM → $\langle x, x \rangle$, for all real numbers x
IDENT → $\langle s, \texttt{"}s\texttt{"} \rangle$, for all possible CLANG identifiers s

The purpose of these initial rules is to provide a default translation for all unseen numbers and identifiers.

	GEOQUERY (880 data set)			ROBOCUP		
	Prec. (%)	*Rec.* (%)	*F* (%)	*Prec.* (%)	*Rec.* (%)	*F* (%)
WASP	87.2	74.8	80.5	88.9	61.9	73.0
COCKTAIL	89.9	79.4	84.3	-	-	-
SILT	89.0	54.1	67.3	83.9	50.7	63.2
KRISP	93.3	71.7	81.1	85.2	61.9	71.7
SCISSOR	**95.5**	77.2	85.4	**90.0**	**80.7**	**85.1**
ZC07	**95.5**	**83.2**	**88.9**	-	-	-

Table 3.2: Performance of semantic parsers on the English corpora

3.3.3 Results and Discussion

Table 3.2 shows the performance of WASP on the English corpora with full training data, compared to five other algorithms:[2]

- COCKTAIL (Tang and Mooney, 2001), a shift-reduce parser based on inductive logic proramming.
- SILT (Kate et al., 2005), a deterministic parser using tree-to-string transformation rules.
- KRISP (Kate and Mooney, 2006), an SVM-based parser using string kernels.
- SCISSOR (Ge and Mooney, 2006), a combined syntactic-semantic parser with discriminative reranking.
- Zettlemoyer and Collins (2007) (abbreviated as ZC07), a probabilistic parser based on relaxed CCG grammars.

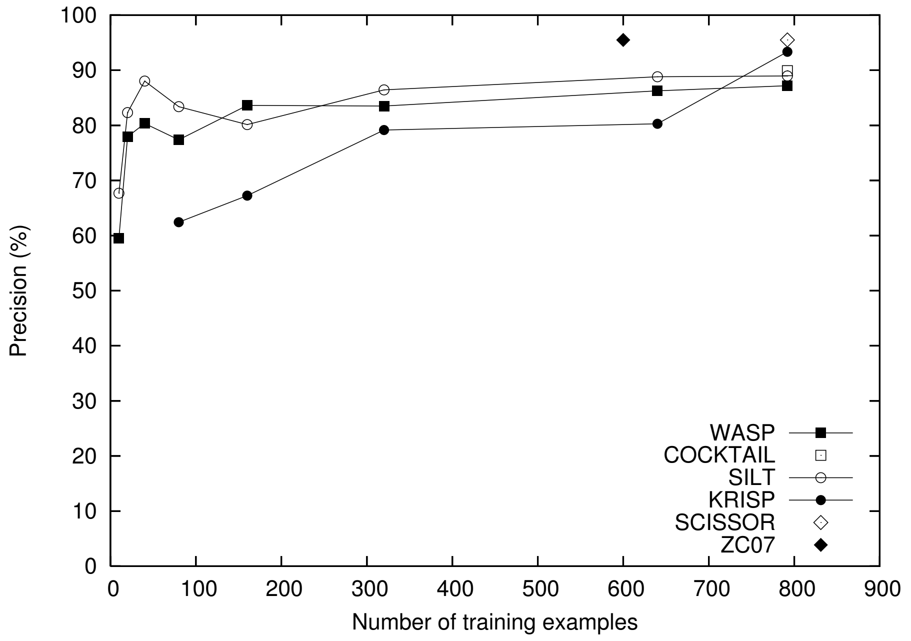
[2]The results reported in Zettlemoyer and Collins (2007) for GEOQUERY are based on a single split of data with 600 training examples. Our experiments using their split gave similar results.

The best-performing systems for each domain are shown in bold in Table 3.2.[3] Figures 3.9 and 3.10 show the precision and recall learning curves.

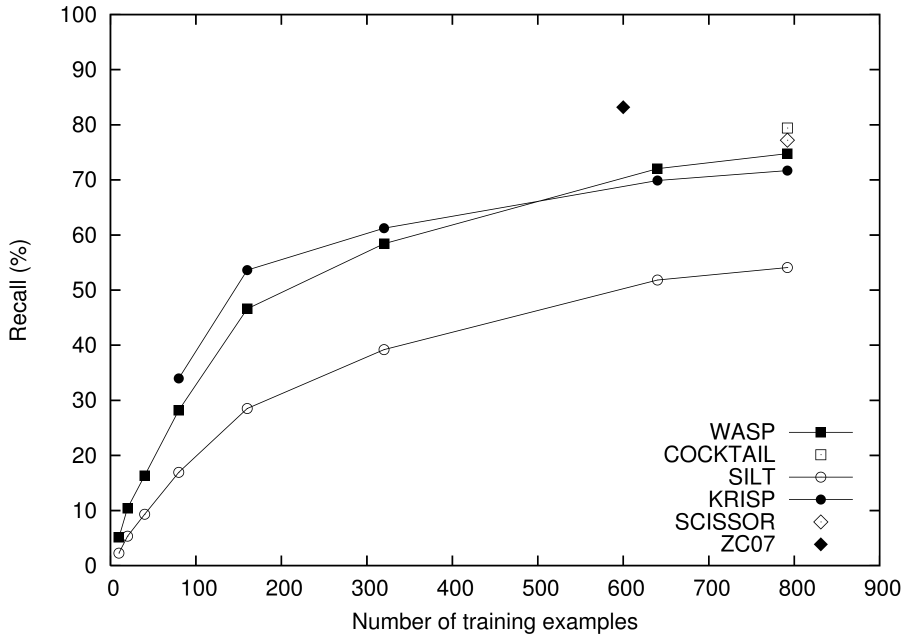
Regarding these results, several points should be noted:

- Due to memory overflow, COCKTAIL cannot handle more than 160 training examples in the ROBOCUP domain.
- No results have been reported for ZC07 in the ROBOCUP domain. In fact, it is unclear how ZC07 can deal with *discontiguous* lexical items which frequently appear in this domain (see Section 5.4.2 for further discussion of discontiguous lexical items).
- Both COCKTAIL and ZC07 use Prolog logical forms as the target MRL for GEOQUERY. In Section 4.3.2, we show that Prolog logical forms can be a better MRL for this domain.
- A hand-built lexicon was supplied to COCKTAIL in the GEOQUERY domain. For ROBOCUP, lexicons automatically acquired by WOLFIE (Thompson and Mooney, 1999) were used instead.
- SCISSOR requires semantically-augmented parse trees for training (Section 2.2.1).
- ZC07 requires the following hand-written components: (1) language-specific template rules (Section 2.2.1), and (2) lexical items for certain function words such as *wh*-words and determiners.

[3]No statistical test was performed for two reasons. First, the experimental set-up in ZC07 was different. Also for SCISSOR, neither per-trial statistics nor actual system output was available.

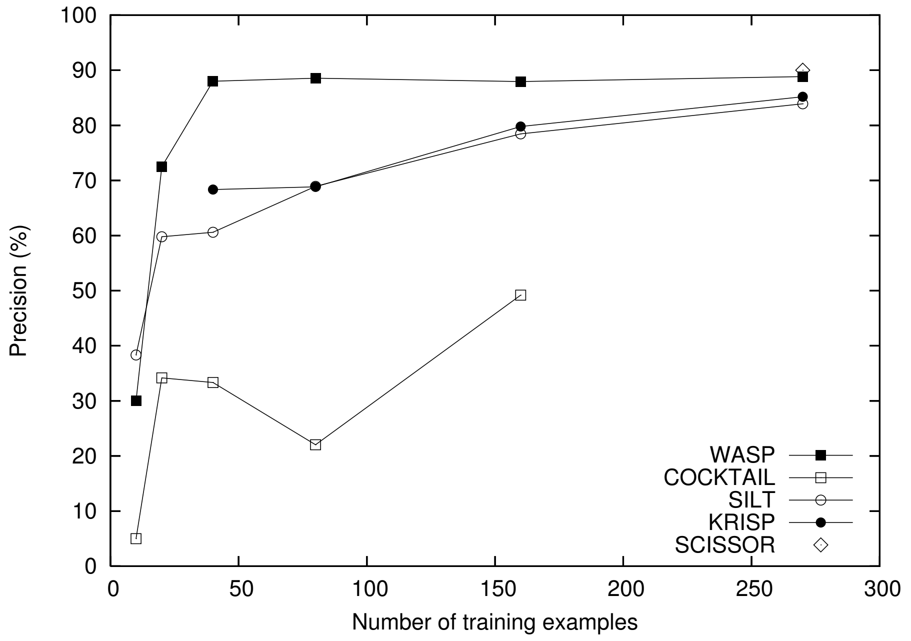


(a) Precision

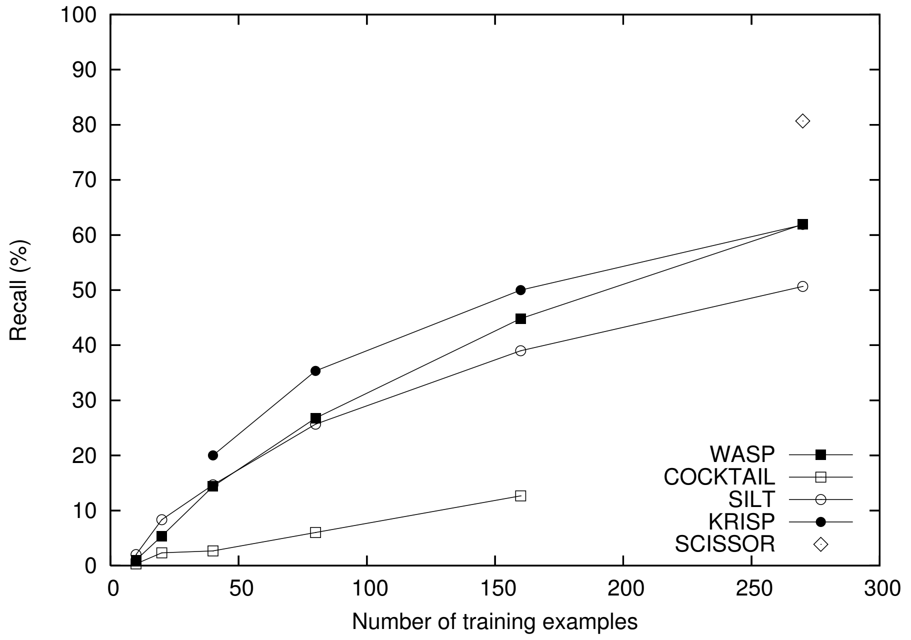


(b) Recall

Figure 3.9: Learning curves for semantic parsers on the GEOQUERY 880 data set



(a) Precision



(b) Recall

Figure 3.10: Learning curves for semantic parsers on the ROBOCUP data set

Therefore, compared to WASP, SILT and KRISP, more prior knowledge was incorporated into COCKTAIL (in the GEOQUERY domain), SCISSOR and ZC07. The experimental results show a clear advantage of such extra supervision, especially in the ROBOCUP domain where sentences are long and data is scarce.

COCKTAIL has very low precision and recall in the ROBOCUP domain. The difficulty apparently lies in the length of sentences being processed. COCKTAIL's deterministic shift-reduce framework processes a sentence only from the beginning to the end. If it fails to parse the beginning of a sentence, then it will fail to parse the rest of the sentence. In contrast, WASP's chart parsing algorithm takes a holistic view of a sentence and is very efficient.

WASP also outperforms SILT in terms of recall. In SILT, transformation rules are learned for each MRL production individually, and the learned rules do not necessarily cooperate to give a complete parse of a training sentence. In WASP, an extracted SCFG always covers the entire training set.

WASP's performance is competitive compared to KRISP. Moreover, as shown in the learning curves, a WASP parser is consistently more precise than a KRISP parser when trained on small data sets.

Overall, our experiments show that WASP performs competitively compared to other methods requiring similar supervision, and is considerably more robust than methods based on deterministic parsing (e.g. COCKTAIL).

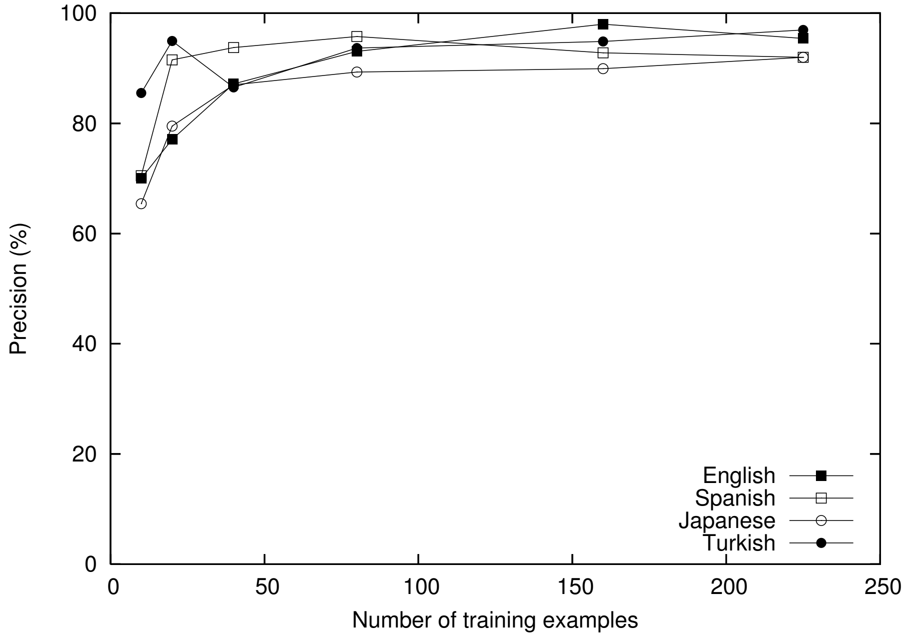
A major advantage of WASP over methods such as SCISSOR and ZC07 is that it does not require any prior knowledge of the NL syntax for training. Therefore, porting WASP to another NL is relatively easy. We illustrate this by evaluating WASP's performance on the multilingual GEOQUERY 250 data set. The languages being considered are English, Spanish, Japanese and Turkish. These languages

	WASP		
	Prec. (%)	*Rec.* (%)	*F* (%)
English	95.42	70.00	80.76
Spanish	91.99	72.40	81.03
Japanese	91.98	74.40	82.86
Turkish	96.96	62.40	75.93

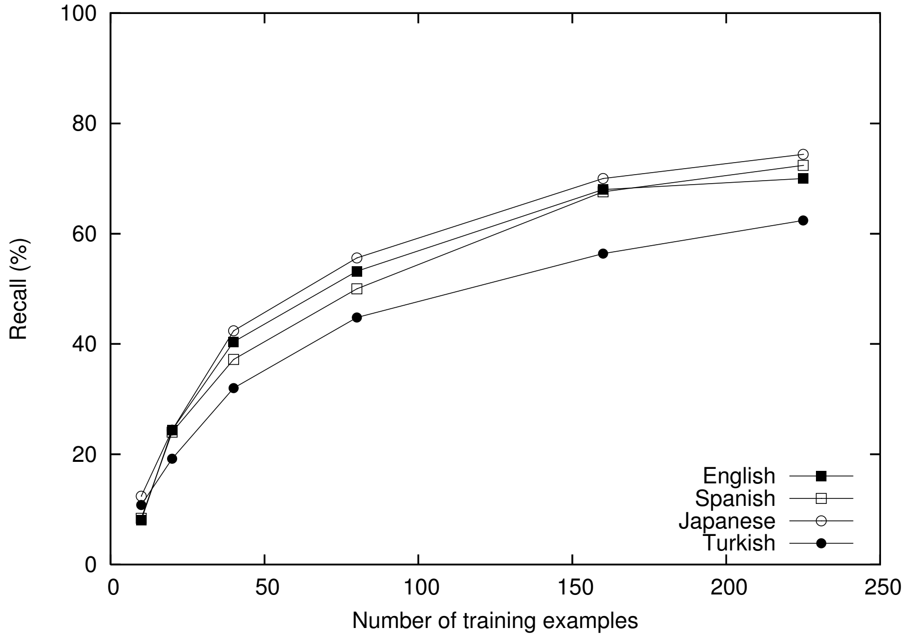
Table 3.3: Performance of WASP on the multilingual GEOQUERY data set

differ in terms of word order: Subject-Verb-Object (SVO) for English and Spanish, and Subject-Object-Verb (SOV) for Japanese and Turkish. They also differ in terms of morphology: English and Spanish are inflected languages, while Japanese and Turkish are agglutinative languages, where words are formed by joining many morphemes together. Each combination of morphemes creates a different word, so agglutinative languages tend to have a larger vocabulary. As shown in Table 3.3 and Figure 3.11, WASP's performance is consistent across all four languages, although recall is lower for Turkish. The reason is that the Turkish corpus has a larger vocabulary (Table 3.1), and the extracted rules tend to be less general. A possible solution is to split words into morphemes and treat each morpheme as a separate token. This has been done by hand for the Japanese corpus.

WASP has much room for improvement compared to methods like SCISSOR and ZC07. The performance gap can be closed by using word alignments derived from the augmented parse trees used for training SCISSOR, in place of the automatic word alignments given by GIZA++ (Table 3.4). This form of extra supervision is shown to improve the precision and recall of WASP slightly. However, we also found that the choice of MRL plays an important role in semantic parsing. In particular, for the GEOQUERY domain, Prolog logical forms can be a more appropriate MRL than FUNQL. In the next chapter, we will explore ways to extend WASP



(a) Precision



(b) Recall

Figure 3.11: Learning curves for WASP on the multilingual GEOQUERY data set

	GEOQUERY (880 data set)			ROBOCUP		
	Prec. (%)	*Rec.* (%)	*F* (%)	*Prec.* (%)	*Rec.* (%)	*F* (%)
WASP	87.2	74.8	80.5	88.9	61.9	73.0
+ hand-written word alignments	93.6	74.1	82.7	**94.6**	65.0	76.8
SCISSOR	**95.5**	77.2	85.4	90.0	**80.7**	**85.1**
ZC07	**95.5**	**83.2**	**88.9**	-	-	-

Table 3.4: Performance of WASP with extra supervision

to handle Prolog logical forms. The resulting algorithm, λ-WASP, uses the same amount of supervision as WASP, and is shown to perform comparably to ZC07, and better than SCISSOR.

3.4 Related Work

The lexical acquisition algorithm of WASP can be seen as projecting syntactic structures from the target MRL to the source ML via an automatically word-aligned parallel corpus. Besides semantic parsing and syntax-based MT, the idea of *structural projection* has also been used for inducing part-of-speech taggers and noun phrase bracketers (Yarowsky and Ngai, 2001), and automating the annotation of less-studied languages (Xia and Lewis, 2007; Moon and Baldridge, 2007).

The problem of phrasal coherence (Section 3.2.3) has recently caught the attention of the statistical MT community (e.g. Fox, 2002). Several syntax-aware word alignment models have been proposed: Cherry and Lin (2006) propose a discriminative word alignment model using features derived from a synchronous grammar. Training this model, however, requires a small set of hand-written word alignments. DeNero and Klein (2007) present a variant of the HMM alignment model (Vogel et al., 1996) with a syntax-sensitive distortion probability distribution. Unlike Cherry and Lin (2006), their method does not require hand-written word alignments for training. In May and Knight (2007), word alignments are made coherent by re-aligning the training set with a learned syntax-based translation model. Both DeNero and Klein's (2007) and May and Knight's (2007) methods can be used in place of the algorithm described in Section 3.2.3 for better performance of WASP.

3.5 Chapter Summary

In this chapter, we formulated the semantic parsing problem as a language translation task, where NL sentences are translated into formal MRs through synchronous parsing. We described a learning algorithm for semantic parsing called WASP. The input to the learning algorithm is a set of training sentences coupled with their correct MRs, and an unambiguous CFG of the target MRL, which is assumed to be variable-free. The output from the learning algorithm is an SCFG, together with parameters that define a log-linear distribution over parses under this grammar. Lexical acquisition is performed using off-the-shelf word alignment models. Since WASP does not require any prior knowledge of the NL syntax for training, porting WASP to other NLs is relatively easy. Experiments showed that WASP's performance is consistent across different languages and domains, and is competitive compared to the currently best methods requiring similar supervision.

Chapter 4

Semantic Parsing with Logical Forms

Formal semantic analysis of natural languages typically uses predicate logic as the representation language. In this chapter, we extend the WASP semantic parsing algorithm to handle logical forms (Wong and Mooney, 2007b). The resulting algorithm, λ-WASP, is based on an extended version of SCFG in which logical forms are generated using the lambda calculus. It is shown to be one of the best-performing systems so far in the GEOQUERY domain.

4.1 Motivation

Traditionally, linguists have used predicate logic to represent meanings associated with NL expressions (Montague, 1970; Dowty et al., 1981). There are many different kinds of predicate logic that deal with different linguistic phenomena such as quantification, modality, underspecification, and discourse. A common feature of these logical languages is the use of *logical variables* to denote entities. For example, in Figure 4.1, the logical variables x_1 and x_2 are used to denote a state and the area of a state, respectively.

In the last chapter, we showed that semantic parsing can be cast as a machine translation task, where an SCFG is used to model the translation of an NL into a formal MRL. But the use of SCFG for semantic parsing is limited to variable-free MRLs, because SCFG does not have a principled mechanism for handling logical

`answer(`x_1`,smallest(`x_2`,(state(`x_1`),area(`x_1`,`x_2`))))`
What is the smallest state by area?

Figure 4.1: A Prolog logical form in GEOQUERY and its English gloss

variables. This is unfortunate because most existing work on computational semantics is based on predicate logic (Charniak and Wilks, 1976; Blackburn and Bos, 2005). For some domains, this problem can be avoided by transforming a logical language into a variable-free, functional language such as FUNQL in GEOQUERY. However, development of such a functional language is non-trivial, and as we will see, logical forms can improve generalization for semantic analysis.

On the other hand, most existing methods for mapping NL expressions to logical forms involve substantial hand-written components that are difficult to maintain. For example, Crouch (2005) describes a semantic interpreter based on a broad-coverage, hand-written lexical functional grammar (Riezler et al., 2002). The English Resource Grammar (Copestake and Flickinger, 2000) is another human-engineered, semantically-grounded grammar which has been used in transfer-based spoken language translation. Other systems contain a hand-written rule-based component that transforms syntactic derivations into semantic representations (Bayer et al., 2004; Bos, 2005). Compared to these systems, the CCG-based parsers by Zettlemoyer and Collins (2005, 2007) are much easier to maintain, but they still rely on a small set of hand-written template rules for generating lexical entries, which can create a knowledge-acquisition bottleneck.

In this chapter, we show that the synchronous parsing framework can be used to translate NL sentences into logical forms. We extend the SCFG formalism by adding variable-binding λ-operators to the MR strings. Complete logical forms are then generated with the lambda calculus (Church, 1940), which is commonly used to provide a compositional semantics for NLs (Montague, 1970; Steedman, 2000). We call the extended grammar formalism a λ-SCFG. We propose a learning algorithm similar to WASP, which learns a λ-SCFG from a set of training sentences paired with their correct logical forms, together with parameters that define a log-linear distribution over parses under the λ-SCFG. We call the extended algorithm λ-WASP. Experiments show that λ-WASP is currently one of the best-performing semantic parsing algorithms in the GEOQUERY domain.

4.2 The λ-WASP Algorithm

This section describes the λ-WASP algorithm. We first define the λ-SCFG formalism (Section 4.2.1). Then we introduce the basic learning algorithm of λ-WASP (Sections 4.2.2 and 4.2.3). While reasonably effective, it can be further improved through transformation of logical forms (Section 4.2.4) and language modeling (Section 4.2.5).

4.2.1 The λ-SCFG Formalism

To see why it is problematic to use an SCFG to generate logical forms, consider the formal query in Figure 4.1. The answer to this formal query, which is the state with the smallest area, is denoted by x_1. Accordingly, x_1 occurs three times in this logical form: the first time under the `answer` predicate, the second time under `state`, and the third time under `area`. An SCFG would generate these three instances of x_1 in three separate steps (Figure 4.2). However, it is very difficult to model their dependencies because of the context-free assumption of an SCFG. What this grammar lacks is a principled mechanism for naming logical variables.

To make it possible to model the dependencies between logical variables, we introduce variable-binding λ-operators to the SCFG formalism. We call the

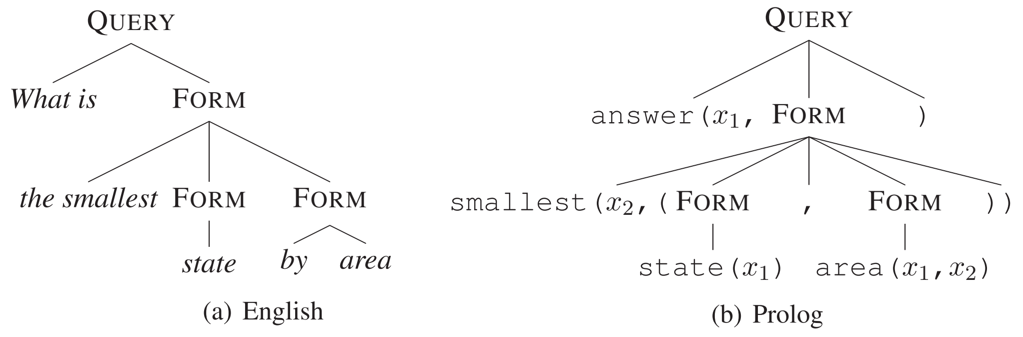


Figure 4.2: An SCFG parse for the string pair in Figure 4.1

resulting grammar a λ-SCFG.

Recall that in an SCFG, each rule has the following form:

$$A \rightarrow \langle \alpha, \beta \rangle \tag{4.1}$$

where A is a non-terminal, α is an NL string, and β is an MRL translation of α. Both α and β are strings of terminal and non-terminal symbols. In a λ-SCFG, each rule has the following form:

$$A \rightarrow \langle \alpha, \lambda x_1 \dots \lambda x_k . \beta \rangle \tag{4.2}$$

where α is an NL string and β is an MRL translation of α. Unlike (4.1), β is a string of terminals, non-terminals, and *logical variables*. The variable-binding operators λ bind occurrences of the logical variables $x_1, \dots, x_k$ in β, and make $\lambda x_1 \dots \lambda x_k . \beta$ a λ-*function* of arity k. When applied to a list of arguments, $(x_{i_1}, \dots, x_{i_k})$, the λ-function gives $\beta\sigma$, where σ is a substitution, $\{x_1/x_{i_1}, \dots, x_k/x_{i_k}\}$, that replaces all bound occurrences of x_j in β with x_{i_j}. For example, in the following expression:

$$(\lambda x_1 . \lambda x_2 . \texttt{area}(x_1, x_2))(x_2, x_3)$$

$\lambda x_1.\lambda x_2.\texttt{area(}x_1\texttt{,}x_2\texttt{)}$ is a λ-function of arity 2 in which occurrences of x_1 and x_2 are bound. When applied to (x_2, x_3), this λ-function gives $\texttt{area(}x_2\texttt{,}x_3\texttt{)}$.

To avoid accidental binding of variables, if any of the arguments x_{i_j} appear in β as a free variable (i.e. not bound by any λ-operators), then those free variables in β must be renamed before function application takes place. For example, in the following λ-function:

$$\lambda x_1.\texttt{(state(}x_1\texttt{),next\_to(}x_1\texttt{,}x_2\texttt{),equal(}x_2\texttt{,stateid(texas)))}$$

x_2 is a free variable. It must be renamed to something other than x_2 so that this λ-function can be applied to (x_2).

Each non-terminal A_j in β is followed by a list of k_j arguments (k_j can be 0). During parsing, A_j must be rewritten by a λ-function of arity k_j. As with an SCFG, a derivation starts with a pair of associated start symbols, and it ends when all non-terminals have been rewritten. For example, Figure 4.3 shows a possible λ-SCFG parse of the string pair in Figure 4.1. The yield of the MR parse tree (Figure 4.3(b)) is the following expression:

```
answer(x1,
      (λx1.smallest(x2,(
            (λx1.state(x1))(x1),
            (λx1.λx2.area(x1,x2))(x1,x2)
      )))(x1)
)
```

Each λ-function in this expression is then applied to its corresponding arguments in a bottom-up manner, resulting in an MR string free of λ-operators with logical variables properly named. For example, given the above expression, we first apply $\lambda x_1.\texttt{state(}x_1\texttt{)}$ to (x_1), and $\lambda x_1.\lambda x_2.\texttt{area(}x_1\texttt{,}x_2\texttt{)}$ to (x_1, x_2). This results

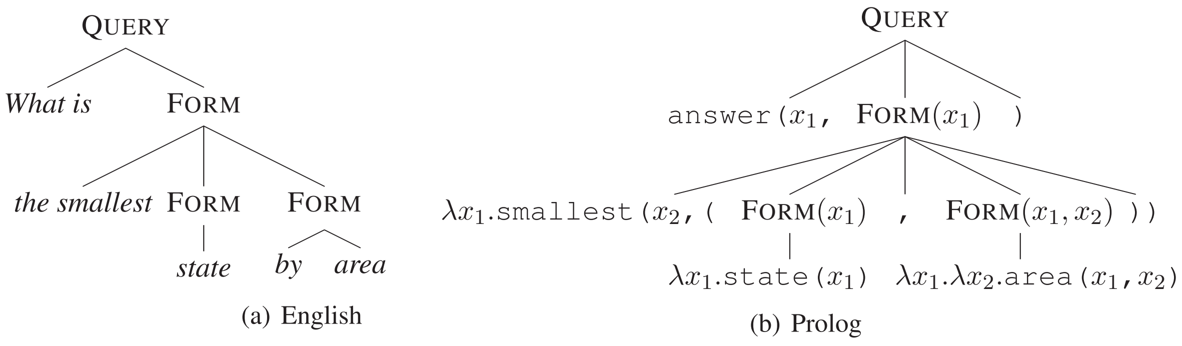


Figure 4.3: A λ-SCFG parse for the string pair in Figure 4.1

in two MR strings: `state(`x_1`)` and `area(`x_1`,`x_2`)`. These two strings combine with λx_1`.smallest(...)` to form a larger λ-function, which is then applied to (x_1). The resulting string, `smallest(...)`, combines with `answer(`x_1`, ...)`, giving the logical form in Figure 4.1.

The following rules can be used to produce the parse trees in Figure 4.3:

$$\begin{aligned}
\textsc{Query} &\rightarrow \langle\ \textit{what is}\ \textsc{Form}_{\boxed{1}}\ ,\ \texttt{answer(}x_1\texttt{,}\textsc{Form}_{\boxed{1}}(x_1)\texttt{)}\ \rangle \\
\textsc{Form} &\rightarrow \langle\ \textit{smallest}\ \textsc{Form}_{\boxed{1}}\ \textsc{Form}_{\boxed{2}}\ , \\
&\qquad \lambda x_1.\texttt{smallest(}x_2\texttt{, (}\textsc{Form}_{\boxed{1}}(x_1)\texttt{,}\textsc{Form}_{\boxed{2}}(x_1,x_2)\texttt{))}\ \rangle \\
\textsc{Form} &\rightarrow \langle\ \textit{state}\ ,\ \lambda x_1.\texttt{state(}x_1\texttt{)}\ \rangle \\
\textsc{Form} &\rightarrow \langle\ \textit{by area}\ ,\ \lambda x_1.\lambda x_2.\texttt{area(}x_1\texttt{,}x_2\texttt{)}\ \rangle
\end{aligned}$$

Note that non-terminals in the NL and MR strings in each rule are indexed with $\boxed{1}, \boxed{2}, \ldots$ to show their association. With the λ-operators, now we have a principled mechanism for naming logical variables across a derivation.

As a side note, the first two rules listed above can be reformulated as follows:

$$\begin{aligned}
\textsc{Query} &\rightarrow \langle\ \textit{what is}\ \textsc{Form}_{\boxed{1}}\ ,\ \lambda p_1.\texttt{answer(}x_1\texttt{,}p_1(x_1)\texttt{)}\ \rangle \\
\textsc{Form} &\rightarrow \langle\ \textit{smallest}\ \textsc{Form}_{\boxed{1}}\ \textsc{Form}_{\boxed{2}}\ , \\
&\qquad \lambda p_1.\lambda p_2.\lambda x_1.\texttt{smallest(}x_2\texttt{, (}p_1(x_1)\texttt{,}p_2(x_1,x_2)\texttt{))}\ \rangle
\end{aligned}$$

In other words, non-terminals in the MR strings can be seen as bound occurrences of logical variables, p_i, that abstract over λ-functions (Dowty et al., 1981, pp. 102–103). The names of these logical variables correspond to the non-terminal indices in the NL strings. This notation of higher-order abstraction has been widely used in the linguistics literature (e.g. Steedman, 2000). However, in this thesis, we will follow our synchronous grammar formulation of λ-SCFG. The reason is two-fold:

1. The synchronous grammar formulation makes explicit the similarity between SCFGs and λ-SCFGs. For example, a λ-SCFG degenerates into an SCFG if none of its rules contain any occurrences of logical variables. As we will see, the lexical acquisition algorithms for SCFGs and λ-SCFGs are also very similar.

2. Compared to logical variables, non-terminals can provide additional, domain-specific type constraints.

We also note that a λ-SCFG can be seen as a generalized context-free grammar (GCFG) (Pollard, 1984), a formalism used by Weir (1988) to characterize mildly context-sensitive grammars. A GCFG is context-free in the sense that rewriting choices in a derivation are independent of the derivation history. It can be shown that this is the case for a λ-SCFG.

The λ-SCFG formalism is also close to LINGOL (Pratt, 1973), a linguistically-oriented programming language designed for NLP tasks such as machine translation and semantic parsing. A LINGOL program can be seen as a synchronous grammar, in which each grammar rule consists of an NL phrase coupled with an arbitrary LISP S-expression (i.e. a small program). These S-expressions combine to produce an analysis of a complete sentence. In the λ-SCFG formalism, such expressions are restricted to be λ-functions which combine solely through function application.

```
answer(x1,count(x2,(state(x2),next_to(x2,x3),
      most(x3,x4,(state(x3),next_to(x3,x4),state(x4)))),x1))
```
How many states border the state that borders the most states?

Figure 4.4: A Prolog logical form in GEOQUERY and its English gloss

4.2.2 Lexical Acquisition

Given a set of training sentences paired with their correct logical forms, $\{\langle \mathbf{e}_i, \mathbf{f}_i \rangle\}$, the first learning task of λ-WASP is to find a λ-SCFG, G, that covers the training data. Like WASP, we construct G using rules extracted from word alignments. We illustrate this using Figures 4.4–4.7. The parse tree in Figure 4.5 is obtained using an unambiguous CFG for Prolog logical forms.[1] In this grammar, each production corresponds to a formula. Also a conjunction operator (`,`) always combines with its left conjunct to avoid ambiguity in the Prolog grammar. Figure 4.6 shows a sample word alignment from which λ-SCFG rules can be extracted using the algorithm described in Sections 3.2.1–3.2.3.

However, this results in a λ-SCFG where logical variables are never bound. Basically, λ-SCFG rules should be extracted from a word alignment based on an MR parse tree where logical variables are explicitly bound by λ-operators (Figures 4.7 and 4.8).

The transformation from Figure 4.5 to Figure 4.7 is straightforward. It can be done in a bottom-up manner, starting with MRL productions with no non-terminals on the RHS, e.g. FORM $\rightarrow$ `state(`x_4`)`. For each production $A \rightarrow \beta$, a logical variable x_i is bound whenever x_i appears in β as well as outside the MR sub-parse rooted at A. Such logical variables need to be bound because otherwise,

[1]Although we focus on Prolog logical forms, techniques developed in this chapter should be applicable to many other logical languages.

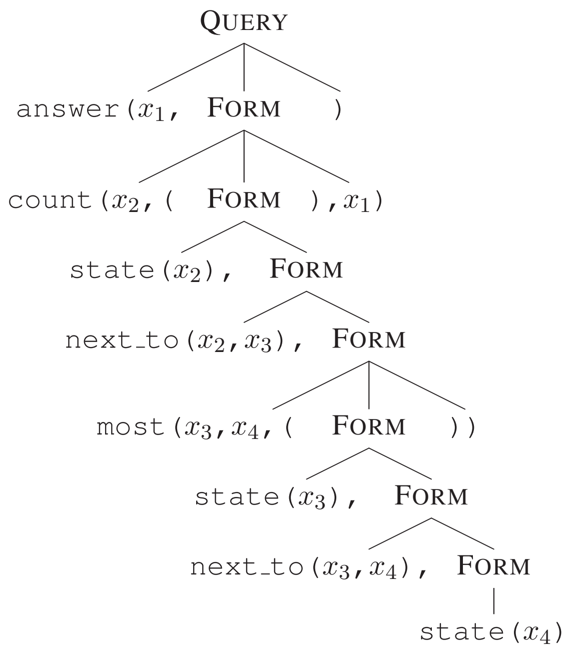
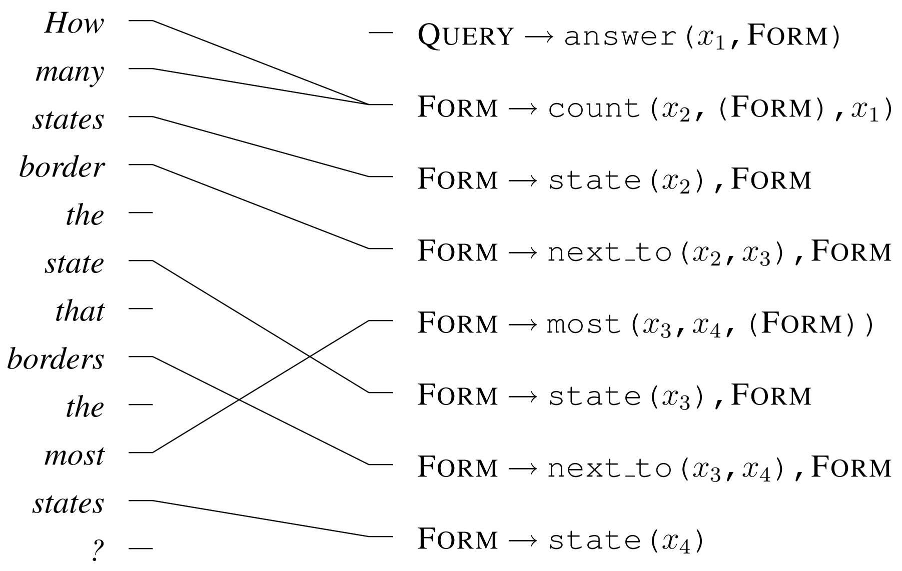

QUERY
answer(x_1, FORM)
count(x_2,(FORM),x_1)
state(x_2), FORM
next_to(x_2,x_3), FORM
most(x_3,x_4,(FORM))
state(x_3), FORM
next_to(x_3,x_4), FORM
state(x_4)

Figure 4.5: A parse tree for the logical form in Figure 4.4

they would be renamed during function application, and therefore, become invisible to the rest of the logical form. Other logical variables need not be bound, e.g. those that only appear in β but not outside. As we add λx_i to β, we also add x_i to the argument list that follows A in the parent MRL production. For example, in Figure 4.5, the logical variable x_4 in FORM $\rightarrow$ `state(`x_4`)` needs to be bound because x_4 appears under the `most` and `next_to` predicates as well. It would also be added to the argument list that follows FORM in the parent MRL production, resulting in FORM $\rightarrow$ `next_to(`x_3`,`x_4`)`,FORM(x_4). This procedure continues upward until the root of the MR parse tree is reached.

Once transformed parse trees are obtained for all logical forms in the training set, lexical acquisition proceeds as follows: Train a word alignment model, M, and a reverse word alignment model, $\tilde{M}$, using the training set, $\{\langle \mathbf{e}_i, \mathbf{f}'_i \rangle\}$, where $\mathbf{f}'_i$



How
many
states
border
the
state
that
borders
the
most
states
?
QUERY → answer(x_1,FORM)
FORM → count(x_2,(FORM),x_1)
FORM → state(x_2),FORM
FORM → next_to(x_2,x_3),FORM
FORM → most(x_3,x_4,(FORM))
FORM → state(x_3),FORM
FORM → next_to(x_3,x_4),FORM
FORM → state(x_4)

Figure 4.6: A word alignment based on Figures 4.4 and 4.5

are the transformed MR parse trees. During the training of M and $\tilde{M}$, all lambda abstractions and variable names in logical forms are ignored to reduce sparsity. Obtain the k-best alignments, $\mathbf{a}^{\star}_{1,\ldots,k}$, and the best reverse alignment, $\tilde{\mathbf{a}}^{\star}$, for each training example $\langle \mathbf{e}_i, \mathbf{f}'_i \rangle$ using M and $\tilde{M}$. Remove bad links from each $\mathbf{a}^{\star}_{k'}$ and replenish the removed links by adding links from $\tilde{\mathbf{a}}^{\star}$ (Section 3.2.3). Then extract λ-SCFG rules from $\mathbf{a}^{\star}_{1,\ldots,k}$ as described in the ACQUIRE-LEXICON procedure (Figure 3.6, lines 7–13), while merging nodes in the MR parse tree if necessary (Section 3.2.2). The extracted λ-functions can be normalized through renaming of logical variables, using a procedure commonly known as *α-conversion* (Blackburn and Bos, 2005).

4.2.3 Probabilistic Model

Since a learned λ-SCFG can be ambiguous, a probabilistic model is needed for parse disambiguation. In λ-WASP, we use the same log-linear model as WASP

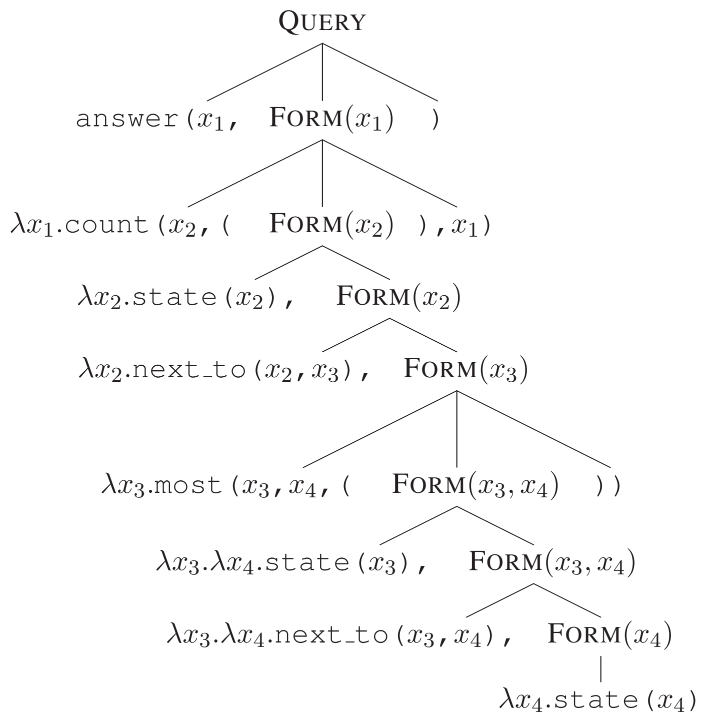

QUERY
answer(x_1,
FORM(x_1)
)
λx_1.count(x_2,(
FORM(x_2)
),x_1)
λx_2.state(x_2),
FORM(x_2)
λx_2.next_to(x_2,x_3),
FORM(x_3)
λx_3.most(x_3,x_4,(
FORM(x_3, x_4)
))
$\lambda x_3.\lambda x_4$.state(x_3),
FORM(x_3, x_4)
$\lambda x_3.\lambda x_4$.next_to(x_3,x_4),
FORM(x_4)
λx_4.state(x_4)

Figure 4.7: A parse tree for the logical form in Figure 4.4 with λ-operators

(Section 3.2.4). In summary, the log-linear model defines a conditional probability distribution over derivations given an input NL sentence, $\mathbf{e}$:

$$\Pr_\lambda(\mathbf{d}|\mathbf{e}) = \frac{1}{Z_\lambda(\mathbf{e})} \exp \sum_i \lambda_i f_i(\mathbf{d}) \tag{4.3}$$

The output logical form, $\mathbf{f}^\star$, is the yield of the most probable derivation consistent with the input sentence, which can be computed in cubic time with respect to the sentence length:

$$\mathbf{f}^\star = \mathbf{f}\left(\arg\max_{\mathbf{d}\in D(G|\mathbf{e})} \exp \sum_i \lambda_i f_i(\mathbf{d})\right) \tag{4.4}$$

The following feature types are used in the log-linear model:

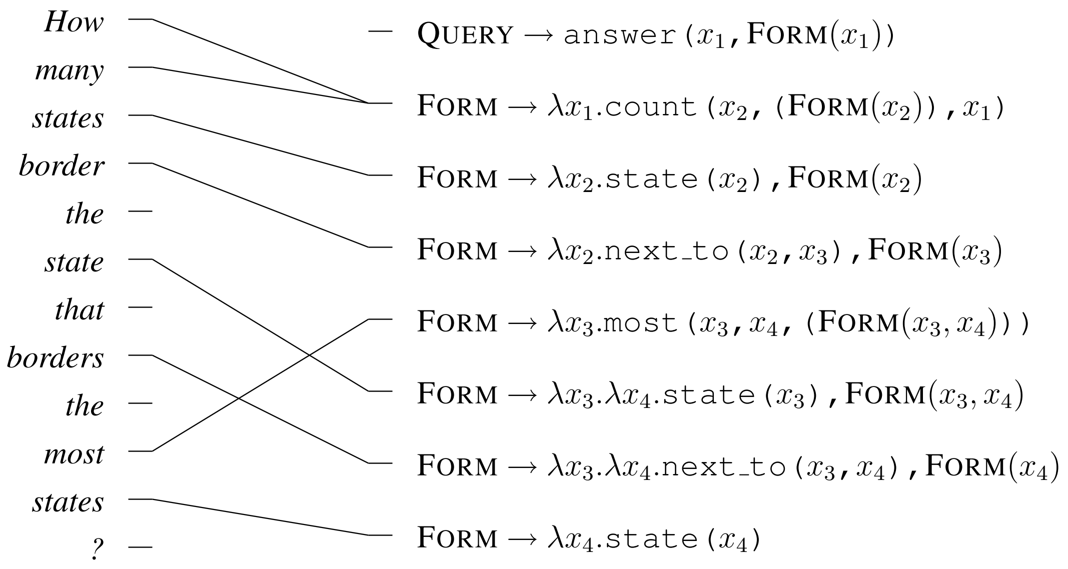

How
many
states
border
the
state
that
borders
the
most
states
?
QUERY → answer(x_1,FORM(x_1))
FORM → λx_1.count(x_2,(FORM(x_2)),x_1)
FORM → λx_2.state(x_2),FORM(x_2)
FORM → λx_2.next_to(x_2,x_3),FORM(x_3)
FORM → λx_3.most(x_3,x_4,(FORM(x_3,x_4)))
FORM → λx_3.λx_4.state(x_3),FORM(x_3,x_4)
FORM → λx_3.λx_4.next_to(x_3,x_4),FORM(x_4)
FORM → λx_4.state(x_4)

Figure 4.8: A word alignment based on Figures 4.4 and 4.7

- For each λ-SCFG rule r, there is a feature, f_r, that returns the number of times r is used in a derivation.
- For each NL word w, there is a feature, f_w, that returns the number of times w is generated from word gaps in a derivation.
- Generation of previously unseen words is modeled using an extra feature, f_*, that returns the total number of words generated from word gaps in a derivation.

Additional language-modeling features specific to λ-WASP will be introduced in Section 4.2.5.

The model parameters, λ, are estimated by maximizing the conditional log-likelihood of the training set.[2] Details of the parameter estimation algorithm can be found in Section 3.2.4.

4.2.4 Promoting Parse Tree Isomorphism

In the previous sections, we have described the λ-WASP algorithm in which logical forms are produced using the lambda calculus. While reasonably effective, it can be further improved in several ways. In this section, we focus on improving lexical acquisition.

To see why the current lexical acquisition algorithm can be problematic, consider the following λ-SCFG rules which would be extracted from the word alignment in Figure 4.8:

$$\begin{aligned}
\textsc{Form} &\rightarrow \langle\ \textit{states}\ ,\ \lambda x_4.\texttt{state(}x_4\texttt{)}\ \rangle\\
\textsc{Form} &\rightarrow \langle\ \textit{state}\ (1)\ \textit{borders}\ (1)\ \textit{most}\ \textsc{Form}_{\boxed{1}}\ ,\\
&\qquad \lambda x_3.\texttt{most(}x_3\texttt{,}x_4\texttt{,(state(}x_3\texttt{),next\_to(}x_3\texttt{,}x_4\texttt{),}\\
&\qquad\qquad \textsc{Form}_{\boxed{1}}(x_4)\texttt{))}\ \rangle\\
\textsc{Form} &\rightarrow \langle\ \textit{border}\ (1)\ \textsc{Form}_{\boxed{1}}\ ,\ \lambda x_2.\texttt{next\_to(}x_2\texttt{,}x_3\texttt{),}\textsc{Form}_{\boxed{1}}(x_3)\ \rangle\\
\textsc{Form} &\rightarrow \langle\ \textit{states}\ \textsc{Form}_{\boxed{1}}\ ,\ \lambda x_2.\texttt{state(}x_2\texttt{),}\textsc{Form}_{\boxed{1}}(x_2)\ \rangle\\
\textsc{Form} &\rightarrow \langle\ \textit{how many}\ \textsc{Form}_{\boxed{1}}\ ,\ \lambda x_1.\texttt{count(}x_2\texttt{,(}\textsc{Form}_{\boxed{1}}(x_2)\texttt{),}x_1\texttt{)}\ \rangle\\
\textsc{Query} &\rightarrow \langle\ \textsc{Form}_{\boxed{1}}\ (1)\ ,\ \texttt{answer(}x_1\texttt{,}\textsc{Form}_{\boxed{1}}(x_1)\texttt{)}\ \rangle
\end{aligned}$$

The second rule is based on the combination of three MRL productions. These productions are combined because no rules can be extracted for the production FORM → $\lambda x_3.\lambda x_4.$`next_to(...)`. This is because the shortest NL substring that covers the word *borders* and the argument string *states*, i.e. *borders the most states*, contains the word *most*, which is linked to an MRL production (`most`) that is not

[2]While the use of the symbol λ for log-linear parameters coincides with the use of λ for variable-binding operators, the meaning of λ should be clear from the context.

a descendent of FORM $\rightarrow \lambda x_3.\lambda x_4.$`next_to(...)` in the MR parse tree. Rule extraction is forbidden in this case because it would destroy the link between *most* and `most`. Same for the production FORM $\rightarrow \lambda x_3.\lambda x_4.$`state(...)`. These two productions are combined with the production for the `most` predicate through node merging (Section 3.2.2). Since excessive node merging can lead to rules that are too specific, causing overfitting, it is desirable to have NL and MR parse trees that are isomorphic, or close to isomorphic.

As mentioned in Section 3.4, several researchers have proposed syntax-aware word alignment models to promote tree isomorphism. Here we use a different approach: change the shape of an MR parse tree so that the NL and MR parse trees are maximally isomorphic. This is possible because the conjunction operator (`,`) used in predicate logic is both associative (`a,(b,c) = (a,b),c = a,b,c`) and commutative (`a,b = b,a`).[3] Hence, conjuncts can be reordered and regrouped without changing the meaning of a conjunction. Such conjunct reordering and regrouping changes the shape of an MR parse tree. For example, rule extraction would be possible if the MR sub-parse for the formula `most(...)` is the one shown in Figure 4.9.

We present a method for regrouping conjuncts to promote isomorphism between NL and MR parse trees. It requires a word alignment as input. This regrouping is done before λ-operators are added (Section 4.2.2). Given a conjunction, it does the following:

Step 1. Identify the MRL productions that correspond to the conjuncts and the predicate that takes the conjunction as an argument, and figure them as vertices in an undirected graph, Γ. For example, in this MR parse tree:

[3]While our discussion focuses on the conjunction operator, it also applies to other operators that are associative and commutative, e.g. disjunction.

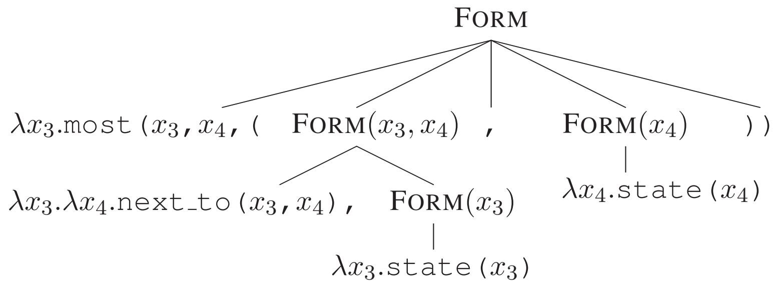
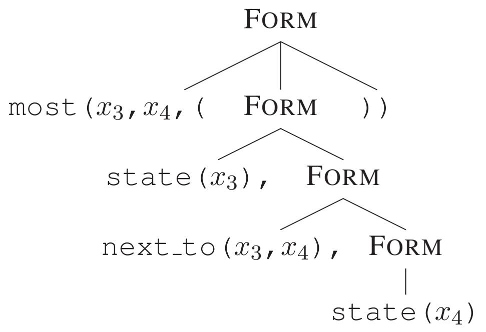


Figure 4.9: An alternative sub-parse for the logical form in Figure 4.4



the productions that correspond to the conjuncts are:

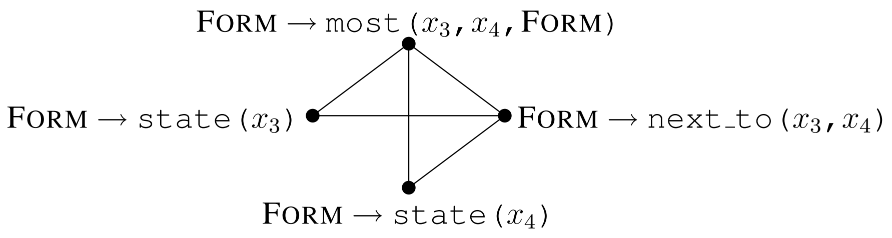
$$\textsc{Form} \rightarrow \texttt{state(}x_3\texttt{)}$$
$$\textsc{Form} \rightarrow \texttt{next\_to(}x_3\texttt{,}x_4\texttt{)}$$
$$\textsc{Form} \rightarrow \texttt{state(}x_4\texttt{)}$$

and the production that corresponds to the predicate that takes the conjunction as an argument is:

$$\textsc{Form} \rightarrow \texttt{most(}x_3\texttt{,}x_4\texttt{,}\textsc{Form}\texttt{)}$$

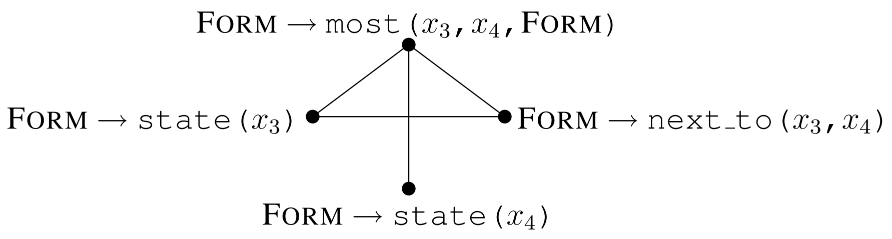
Each of these productions, denoted by p_i, is figured as a vertex in the undirected graph Γ. For convenience, the production that corresponds to the predicate that takes the conjunction as an argument has a special name, p_0.

Step 2. Add an edge (p_i, p_j) to Γ if there exists a logical variable x that appears in the RHS of both p_i and p_j. For example, Γ would look like this:



Each edge in Γ indicates a possible edge in the rearranged MR parse tree. Intuitively, two concepts are closely related only if they involve the same logical variables, and closely-related concepts should be placed close together in the MR parse tree. By keeping occurrences of a logical variable in close proximity in the MR parse tree, we also avoid unnecessary variable bindings in the extracted rules.

Step 3. Let $\mathrm{s}(i, j)$ be the shortest NL substring that contains all the words that are linked to p_i and p_j in the input word alignment. If $i, j \neq 0$ and $\mathrm{s}(i, j)$ contains a word that is linked to p_0, then remove the edge (p_i, p_j) from Γ. For example, Γ would look like this given the word alignment in Figure 4.6:

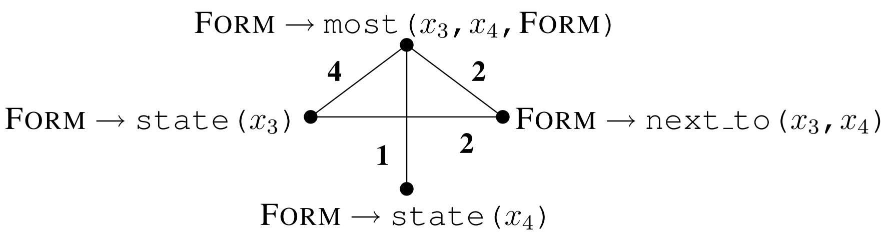


An edge is removed because the shortest NL substring that contains all the words that are linked to FORM → `next_to`(x_3,x_4) and FORM → `state`(x_4), i.e. *borders the most states*, contains the word *most* which is linked to the production

FORM → `most`(x_3`,`x_4`,`FORM`)`. Since FORM → `most`(x_3`,`x_4`,`FORM`)` is going to be the root of the rearranged MR parse tree, an edge between FORM → `next_to`(x_3`,`x_4`)` and FORM → `state`(x_4`)` would prevent a λ-SCFG rule from being extracted for either the `next_to` or `state` production.

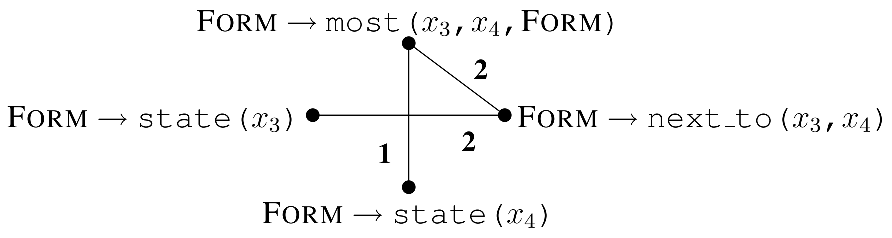
Step 4. To make sure that Γ is a connected graph, add an edge (p_0, p_i) to Γ if p_i is not already connected to p_0 in Γ.

Step 5. Assign edge weights based on word distance. The weight of an edge (p_i, p_j) is defined as the minimum distance between the words that are linked to p_i and p_j. For example, the edge weights for Γ given the word alignment in Figure 4.6 would be:



The weight of the edge between FORM → `most`(x_3`,`x_4`,`FORM`)` and FORM → `state`(x_3`)` is 4 because the words *most* and *state* are 4 words apart in the sentence. The other edge weights are assigned in a similar way.

Step 6. Find a minimum spanning tree, T, for Γ. T exists because Γ is a connected graph (see Step 4). T can be found using Kruskal's algorithm (Cormen et al., 2001). Conjuncts will be regrouped based on T. For example, for the weighted graph Γ shown above, the minimum spanning tree would be:



Conjuncts would be regrouped such that there is an edge in the rearranged MR parse tree between FORM → most(x_3,x_4,FORM) and FORM → next_to(x_3,x_4), and so on. The choice of T reflects the intuition that words that occur close together in a sentence tend to be semantically related.

Step 7. Finally, using p_0 as the root, construct a new MR parse tree based on T. Add conjunction operators to the productions as necessary.

In summary, conjuncts are regrouped such that concepts that are related are placed close together in the MR parse tree (Steps 2, 5 and 6). Also the NL and MR parse trees should be isomorphic if possible (Step 3). This procedure is repeated for all conjunctions that appear in a logical form.

Lexical acquisition then proceeds as described in Section 4.2.2, using the same word alignments used for conjunct regrouping. Figure 4.9 shows the rearranged MR parse tree based on the minimum spanning tree shown above, with λ-operators added. With this MR parse tree, the following λ-SCFG rules would be extracted:

$$\textsc{Form} \rightarrow \langle\ \textit{states}\ ,\ \lambda x_4.\texttt{state(}x_4\texttt{)}\ \rangle$$
$$\textsc{Form} \rightarrow \langle\ \textit{state}\ ,\ \lambda x_3.\texttt{state(}x_3\texttt{)}\ \rangle$$
$$\textsc{Form} \rightarrow \langle\ \textsc{Form}_{\boxed{1}}\ (1)\ \textit{borders}\ ,\ \lambda x_3.\lambda x_4.\texttt{next\_to(}x_3\texttt{,}x_4\texttt{)}\texttt{,}\textsc{Form}_{\boxed{1}}(x_3)\ \rangle$$
$$\textsc{Form} \rightarrow \langle\ \textsc{Form}_{\boxed{1}}\ (1)\ \textit{most}\ \textsc{Form}_{\boxed{2}}\ ,$$
$$\lambda x_3.\texttt{most(}x_3\texttt{,}x_4\texttt{, (}\textsc{Form}_{\boxed{1}}(x_3,x_4)\texttt{,}\textsc{Form}_{\boxed{2}}(x_4)\texttt{))}\ \rangle$$

These rules are considerably shorter than those shown earlier in this section, and therefore would generalize better.

Note that the conjunct regrouping procedure requires a good word alignment to begin with, and this requires a reasonable ordering of conjuncts in the training data, since the word alignment model (GIZA++) is sensitive to word order. This immediately suggests an iterative algorithm in which a better grouping of conjuncts leads to a better alignment model, which is used to guide further regrouping until convergence. We did not pursue this direction, however, because in the restricted domain we worked with, GIZA++ seemed to perform quite well without re-training.

4.2.5 Modeling Logical Languages

In this section, we propose two methods for modeling logical languages. This is motivated by the fact that many of the errors made by the λ-WASP semantic parser can be detected by inspecting the MRL translations alone. Figure 4.10 shows some typical errors, which can be classified into two broad categories:

1. Type mismatch errors. For example, a state cannot possibly be a river (Figure 4.10(a)). Also it is awkward to talk about the population density of a state's highest point (Figure 4.10(b)).

2. Errors that do not involve type mismatch. For example, a query can be overly trivial (Figure 4.10(c)), or involve aggregate functions on a known singleton (Figure 4.10(d)).

The first type of errors can be fixed by type checking. Each m-place predicate is associated with a list of m-tuples showing all valid combinations of entity types that the m arguments can denote:

(a) `answer(`x_1`,largest(`x_2`,(state(`x_1`),major(`x_1`),river(`x_1`),`
`traverse(`x_1`,`x_2`))))`
What is the entity that is a state and also a major river, that traverses something that is the largest?

(b) `answer(`x_1`,smallest(`x_2`,(highest(`x_1`,(place(`x_1`),`
`loc(`x_1`,`x_3`),state(`x_3`))),density(`x_1`,`x_2`))))`
Among the highest points of all states, which one has the lowest population density?

(c) `answer(`x_1`,equal(`x_1`,stateid(alaska)))`
Alaska?

(d) `answer(`x_1`,largest(`x_2`,(largest(`x_1`,(state(`x_1`),`
`next_to(`x_1`,`x_3`),state(`x_3`))),population(`x_1`,`x_2`))))`
Among the largest state that borders some other state, which is the one with the largest population?

Figure 4.10: Typical errors made by λ-WASP with English interpretations

`point(_)` : $\{(\text{POINT})\}$
`density(_,_)` : $\{(\text{COUNTRY}, \text{NUM}), (\text{STATE}, \text{NUM}), (\text{CITY}, \text{NUM})\}$

These m-tuples of entity types are given as domain knowledge.[4] The parser maintains a set of possible entity types for each logical variables introduced in a partial derivation (except those that are no longer visible). If there is a logical variable that cannot denote any type of entity (i.e. its set of entity types is empty), then the partial derivation is considered invalid. For example, based on the tuples shown above, `point(`x_1`)` and `density(`x_1`,_)` cannot be both true, because $\{\text{POINT}\} \cap \{\text{COUNTRY}, \text{STATE}, \text{CITY}\} = \emptyset$. The use of type checking is to exploit

[4]Note that the same entity type information is encoded in the non-terminal symbols in FUNQL (Appendix A.2), so this is not additional domain knowledge compared to what is used in WASP.

the fact that people tend not to ask questions that obviously have no valid answers (Grice, 1975). It is also similar to Schuler's (2003) use of model-theoretic interpretations to guide syntactic parsing.

Errors that do not involve type mismatch are handled by adding new features to the log-linear model (Section 4.2.3). We only consider features that are based on the MRL translations, and therefore, these features can be seen as an implicit language model of the target MRL (Papineni et al., 1997). Of the many feature types that we have tried, one feature type stands out as being the most effective, namely the *two-level rules* in Collins and Koo (2005), which gives the number of times a given rule is used to rewrite a non-terminal in a given parent rule. We use only the MRL part of the rules. For example, a negative weight for the combination of $\text{Query} \rightarrow \texttt{answer(}x_1\texttt{,}\text{Form}(x_1)\texttt{)}$ and $\text{Form} \rightarrow \lambda x_1.\texttt{equal(}x_1\texttt{,\_)}$ would discourage any parse that yields Figure 4.10(c). The two-level-rules features, along with the features described in Section 4.2.3, are used in the final version of λ-WASP.

4.3 Experiments

In this section, we describe our experiments on λ-WASP and analyze the experimental results.

4.3.1 Data Sets and Methodology

We evaluated λ-WASP in the GEOQUERY domain, using the same data set that we used for evaluating WASP (Section 3.3.1). We used the original Prolog logical forms, and Table 4.1 shows the corpus statistics.

We performed standard 10-fold cross validation in our experiments, using precision, recall and F-measure as the evaluation metrics (Equations 3.7–3.9). We

	GEOQUERY				
MRL	Prolog				
# non-terminals	14				
# productions	50				
# examples	250				880
NL	English	Spanish	Japanese	Turkish	English
Avg. sent. length	6.87	7.39	9.11	5.76	7.57
# unique words	165	159	158	220	280

Table 4.1: Corpora used for evaluating λ-WASP

supplied the same set of initial rules to the learned semantic parsers as described in Section 3.3.2. These initial rules represent knowledge needed for translating basic domain entities, such as city names and river names.

4.3.2 Results and Discussion

Table 4.2 shows the performance of λ-WASP on the GEOQUERY 880 data set with full training data, compared to WASP and three other algorithms:

- KRISP (Kate and Mooney, 2006), an SVM-based parser using string kernels.
- SCISSOR (Ge and Mooney, 2006), a combined syntactic-semantic parser with discriminative reranking.
- Zettlemoyer and Collins (2007) (abbreviated as ZC07), a probabilistic parser based on relaxed CCG grammars.

We restrict our comparison to these three algorithms because they were shown to outperform WASP in the GEOQUERY domain in Section 3.3.3. Both λ-WASP and ZC07 use Prolog logical forms as the target MRL. The other systems, WASP, KRISP

	GEOQUERY (880 data set)		
	Prec. (%)	*Rec.* (%)	*F* (%)
λ-WASP	92.0	**86.6**	**89.2**
WASP	87.2	74.8	80.5
KRISP	93.3	71.7	81.1
SCISSOR	**95.5**	77.2	85.4
ZC07	**95.5**	83.2	88.9

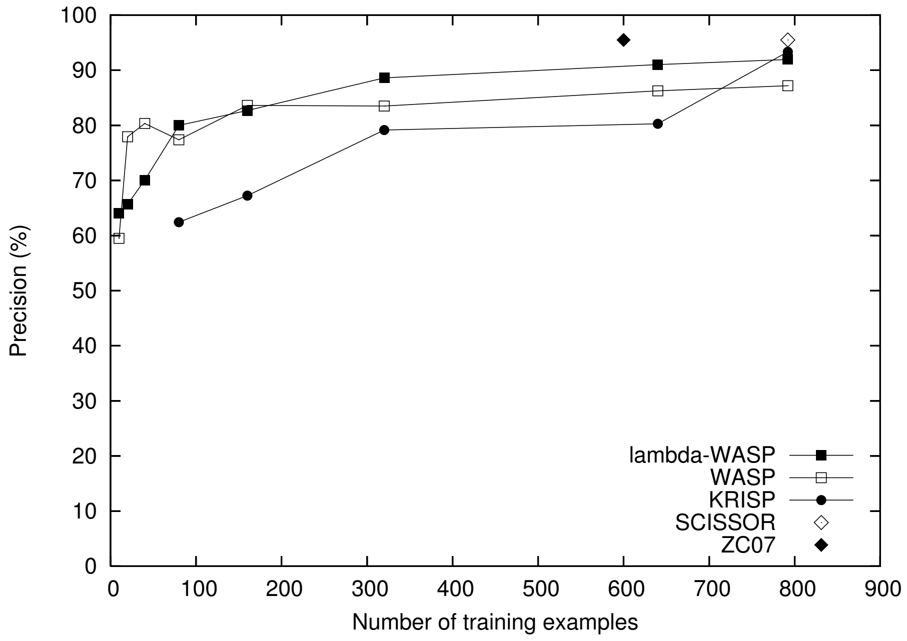
Table 4.2: Performance of λ-WASP on the GEOQUERY 880 data set

and SCISSOR, use the functional query language FUNQL developed for the GEOQUERY domain (Appendix A.2). The best-performing systems are shown in bold in Table 4.2.[5] Figure 4.11 shows the precision and recall learning curves.

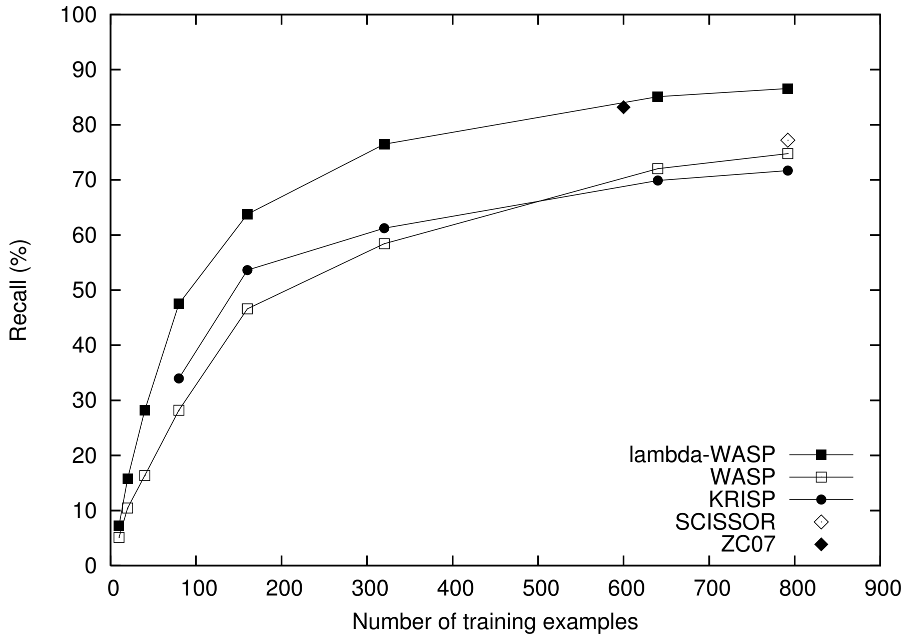
A few observations can be made. First, algorithms that use Prolog logical forms as the target MRL generally show better recall than those using FUNQL. In particular, λ-WASP has the best recall among all systems. The main reason is that λ-WASP allows lexical items to be combined in ways not allowed by FUNQL or the hand-written template rules in ZC07. For example, under FUNQL and ZC07, it is impossible to combine the `most` predicate with its arguments as illustrated in Figure 4.9. Nor is it possible to combine the `smallest` predicate with its arguments as illustrated in Figure 4.3(b). These examples show that λ-WASP is more flexible and can handle a wider variety of logical forms than previous approaches. Despite its slightly lower precision compared to KRISP, SCISSOR and ZC07, λ-WASP has the best F-measure overall in the GEOQUERY domain.

To see the relative importance of each component of the λ-WASP algorithm, we performed two ablation studies. First, we compared the performance of λ-WASP

[5]As with Table 3.2, no statistical test was performed.



(a) Precision



(b) Recall

Figure 4.11: Learning curves for λ-WASP on the GEOQUERY 880 data set

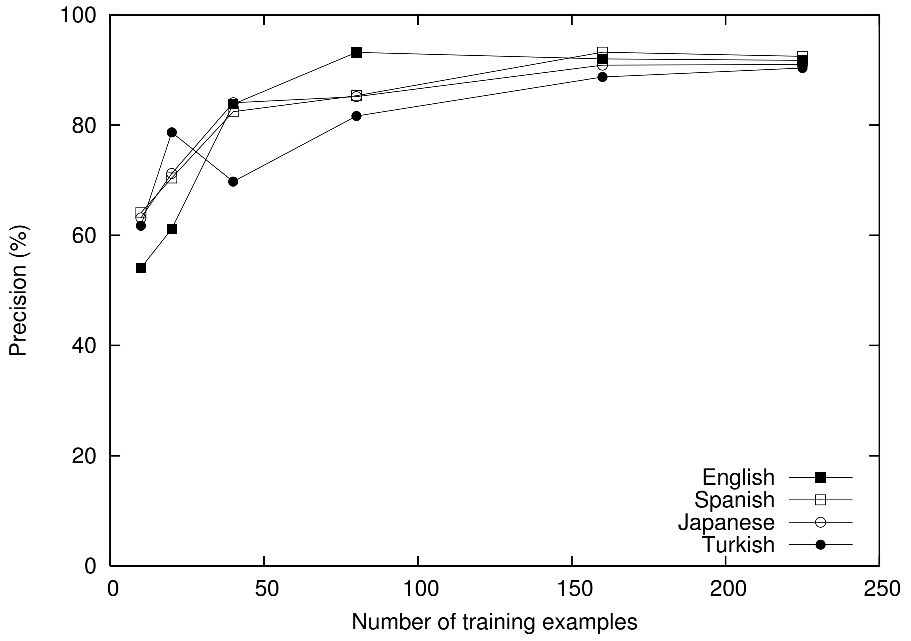
(%)	GEO 880 Prec.	GEO 880 Rec.
λ-WASP	**91.95**	**86.59**
w/o conj. regrouping	**90.73**	83.07

(%)	GEO 880 Prec.	GEO 880 Rec.
λ-WASP	**91.95**	**86.59**
w/o two-level rules	88.46	84.32
and w/o type checking	65.45	63.18

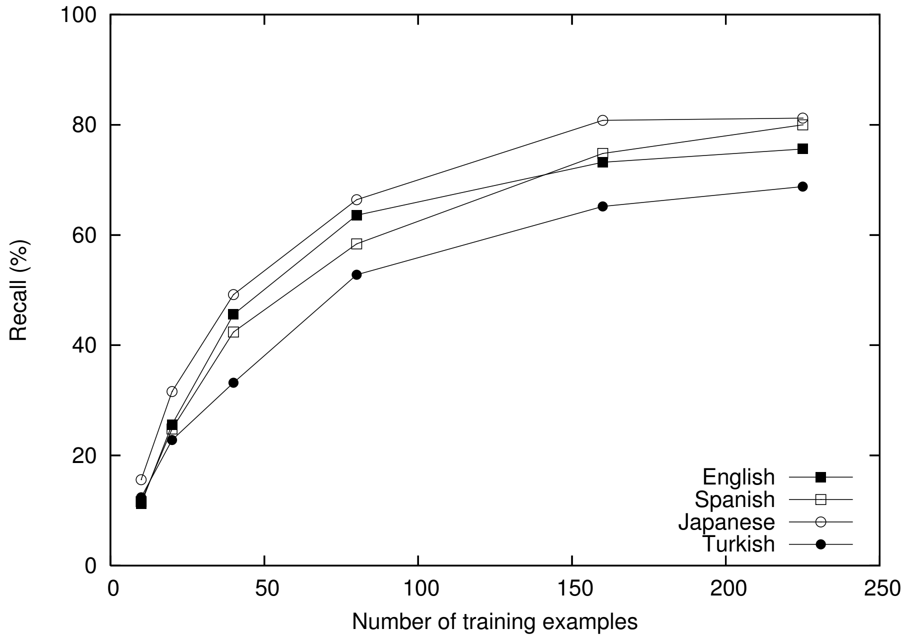
Table 4.3: Performance of λ-WASP with different components removed

with and without conjunct regrouping (Section 4.2.4). Second, we compared the performance of λ-WASP with and without language modeling for the target logical language (Section 4.2.5). Table 4.3 shows the results on the GEOQUERY 880 data set. Using paired t-tests to determine statistical significance, we found that conjunct regrouping improves recall significantly ($p < 0.01$), and the use of *two-level-rules* features in the probabilistic model improves precision and recall ($p < 0.05$). Type checking also significantly improves precision and recall ($p < 0.001$). The best-performing systems, as well as those systems whose performance shows no significant difference, are shown in bold in Table 4.3.

A major advantage of λ-WASP over SCISSOR and ZC07 is that it does not require any prior knowledge of the NL syntax. Hence it is straightforward to apply λ-WASP to other NLs for which training data is available. Table 4.4 shows the performance of λ-WASP on the multilingual GEOQUERY data set. It shows that λ-WASP performed comparably for all four NLs being considered: English, Spanish, Japanese and Turkish. It achieved the same level of precision as WASP (differences are not statistically significant based on paired t-tests). For Spanish and Japanese, λ-WASP has better recall and F-measure than WASP ($p < 0.05$). Figure 4.12 shows the precision and recall learning curves for λ-WASP on the multilingual GEOQUERY data set.



(a) Precision



(b) Recall

Figure 4.12: Learning curves for λ-WASP on the multilingual GEOQUERY data set

	λ-WASP			WASP		
	Prec. (%)	*Rec.* (%)	*F* (%)	*Prec.* (%)	*Rec.* (%)	*F* (%)
English	**91.76**	**75.60**	**82.90**	**95.42**	**70.00**	**80.76**
Spanish	**92.48**	**80.00**	**85.79**	**91.99**	72.40	81.03
Japanese	**90.99**	**81.20**	**85.82**	**91.98**	74.40	82.86
Turkish	**90.36**	**68.80**	**78.12**	**96.96**	**62.40**	**75.93**

Table 4.4: Performance of λ-WASP on the multilingual GEOQUERY data set

4.4 Chapter Summary

In this chapter, we described the λ-WASP semantic parsing algorithm, an extended version of WASP which handles MRLs containing logical variables, such as predicate logic. Underlying λ-WASP is the λ-SCFG formalism, which generates logical forms using the lambda calculus. We described a learning algorithm similar to WASP, whose output is a λ-SCFG, together with parameters that define a log-linear distribution over parses. We further refined the learning algorithm through transformation of logical forms and language modeling for target MRLs. Using the same amount of supervision, λ-WASP significantly outperforms WASP, and is currently one of the best semantic parsing algorithms in the GEOQUERY domain.

Chapter 5

Natural Language Generation with Machine Translation

This chapter explores a different task from semantic parsing, namely natural language generation. We focus on the sub-task of tactical generation, in which statements written in a formal MRL are mapped into NL sentences (Wong and Mooney, 2007a). We show that an effective tactical generation system can be obtained by inverting the WASP semantic parsing algorithm (Chapter 3). Our approach allows the same learned synchronous grammar to be used for both parsing and generation. In this chapter, we consider variable-free MRLs such as FUNQL and CLANG (Section 2.1). Generation from logical forms will be discussed in Chapter 6.

5.1 Motivation

Traditionally, there are several NLP tasks that involve the generation of NL sentences, e.g. natural language generation, machine translation, text summarization, and dialog systems. The goal of *natural language generation* (NLG) is to produce NL sentences from computer-internal representations of information. NLG can be divided into two sub-tasks: (1) *strategic generation*, which decides what meanings to express, and (2) *tactical generation*, which generates NL sentences that express those meanings. This chapter is concerned with the latter task of tactical generation. In this work, we assume that statements written in a formal MRL, produced by an external content planner, are given to a tactical generator as input.

As with NLG, the task of machine translation (MT) involves the generation of NL sentences. NLG is mainly associated with MT in the context of interlingual and transfer-based MT, where an NLG component is used to generate the target language from abstract meaning representations (Wilks, 1973; Nyberg and Mitamura, 1992; Gao et al., 2006). Despite their similar goals, there has been little, if any, research on exploiting recent MT methods for NLG. Specifically, it is easy to use statistical MT to construct a tactical generator, given a corpus of NL sentences coupled with their MRs. In this chapter, we present results on using a recent phrase-based statistical MT system, PHARAOH (Koehn et al., 2003), for NLG. Although moderately effective, the inability of PHARAOH to exploit the formal structure and grammar of the MRL limits its accuracy. Unlike natural languages, MRLs typically have a simple, formal syntax to support effective automated processing and inference. This MRL structure can also be used to improve language generation.

Tactical generation can also be seen as the *inverse* of semantic parsing. In this chapter, we show how to invert the WASP semantic parsing algorithm to produce a more effective generation system. As shown in Chapter 3, WASP exploits the formal syntax of the MRL by learning a translator based on an SCFG that maps an NL sentence to an MR parse tree rather than to a flat MR string. In addition to exploiting the formal MRL grammar, our approach also allows the same learned grammar to be used for both parsing and generation, an elegant property that has been widely advocated (Section 2.3.1). We call our new generation algorithm WASP^{-1}.

While reasonably effective, both PHARAOH and WASP^{-1} can be substantially improved by borrowing ideas from each other. In subsequent sections, we show how the idea of generating from MR parse trees rather than flat MRs, used effectively in WASP^{-1}, can also be exploited in PHARAOH. A version of PHARAOH that exploits this approach is experimentally shown to produce more accurate generators that are more competitive with WASP^{-1}'s. We also show how aspects of PHARAOH's phrase-based model can be used to improve WASP^{-1}, resulting in a hybrid system whose performance is the best.

Overall, we show that effective tactical generation systems can be obtained by exploiting statistical MT methods. This is achieved by treating tactical generation as a language translation task in which formal MRs are *translated* into NL sentences. Furthermore, we show that tactical generation can be formalized as synchronous parsing (Section 2.4), as is the case with semantic parsing and MT.

5.2 Generation with Statistical Machine Translation

In this section, we show how statistical MT methods can be used to construct tactical generators. We first describe a tactical generation algorithm based on PHARAOH, a phrase-based statistical MT system (Section 5.2.1). Then we introduce WASP^{-1} (Section 5.2.2), a tactical generation algorithm which is the inverse of the WASP semantic parsing algorithm.

We consider source MRLs that are variable-free. We also assume that the order in which MR symbols appear is relevant, i.e. the order can affect the meaning of the MR. Note that the order in which MR symbols appear need not be the same as the word order of the target NL, and therefore, the content planner need not know about the target NL grammar (Shieber, 1993).

To ground our discussion, we consider two domains previously used to test WASP's semantic parsing ability, namely GEOQUERY and ROBOCUP (Section 2.1). In the GEOQUERY domain, the task is to translate formal queries into NL queries. Figure 5.1(a) shows a sample formal query and its English translation. In the ROBOCUP domain, the task is to translate formal advice given to soccer-playing

```
answer(state(traverse_1(riverid('ohio'))))
```
What states does the Ohio run through?

(a) A formal query written in FUNQL

```
((bowner our {4}) (do our {6} (pos (left (half our)))))
```
If our player 4 has the ball, then our player 6 should stay in the left side of our half.

(b) A piece of formal advice written in CLANG

Figure 5.1: Sample meaning representations and their English glosses

agents into English. Figure 5.1(b) shows a piece of sample advice and its English translation. Such generation systems can be useful in the *parse disambiguation* scenario: If a semantic parser finds its NL input ambiguous and produces multiple alternative formal interpretations, the competing interpretations can be paraphrased back into NL through a tactical generator, so that the user can pick a correct interpretation based on the NL translations. The chosen interpretation can then be used for further processing. It can also be used as a new training example to improve the semantic parser.

5.2.1 Generation Using PHARAOH

We start with a generation system based on PHARAOH. PHARAOH (Koehn et al., 2003) is a statistical MT system that uses phrases as basic translation units. During decoding, the source sentence is segmented into a number of sequences of consecutive words (or *phrases*). These phrases are then reordered and translated into phrases in the target language, which are joined together to form the output sentence. Compared to earlier word-based methods such as IBM Models (Section 2.5.1), phrase-based methods such as PHARAOH are much more effective in producing idiomatic translations, and are currently among the best-performing methods in statistical MT (Koehn and Monz, 2006).

A main component of PHARAOH is a lexicon consisting of bilingual phrase pairs. These phrase pairs are extracted from a training corpus of sentences coupled with their translations. Using GIZA++, the best word alignments for each training example are first obtained (Section 2.5.1). A lexicon is then formed by collecting all phrase pairs that are consistent with these word alignments.

To discriminate good translations from bad ones, PHARAOH uses a log-linear model that defines a conditional probability distribution over translations:

$$\begin{aligned}\Pr_\lambda(\mathbf{e}|\mathbf{f}) \;\propto\; & \Pr(\mathbf{e})^{\lambda_1} \prod_{i=1}^{I} \big(P(\bar{f}_i|\bar{e}_i)^{\lambda_2} P(\bar{e}_i|\bar{f}_i)^{\lambda_3} P_w(\bar{f}_i|\bar{e}_i)^{\lambda_4} P_w(\bar{e}_i|\bar{f}_i)^{\lambda_5} \\ & d(i-1,i)^{\lambda_6} \exp(-|\bar{e}_i|)^{\lambda_7} \exp(-1)^{\lambda_8} \big) \qquad (5.1)\end{aligned}$$

where $\mathbf{f}$ is an input sentence, and $\mathbf{e}$ is a translation of $\mathbf{f}$. $\Pr(\mathbf{e})$ is the language model. $\bar{e}_i$ and $\bar{f}_i$ are the phrases that comprise $\mathbf{e}$ and $\mathbf{f}$. $P(\bar{e}|\bar{f})$ and $P(\bar{f}|\bar{e})$ are the relative frequencies of $\bar{e}$ and $\bar{f}$, and $P_w(\bar{e}|\bar{f})$ and $P_w(\bar{f}|\bar{e})$ are the lexical weights (Koehn et al., 2003). The distortion model, $d(i,j)$, gives the cost of phrase reordering based on the distance between the i-th and j-th phrases. Both the word penalty, $\exp(-|\bar{e}|)$, and the phrase penalty, $\exp(-1)$, allow some control over the output translation length. The model parameters, λ, are trained using minimum error-rate training (Och, 2003). The output translation, $\mathbf{e}^\star$, given an input sentence, $\mathbf{f}$, is:

$$\mathbf{e}^\star = \arg\max_{\mathbf{e}} \Pr_\lambda(\mathbf{e}|\mathbf{f}) \qquad (5.2)$$

This can be efficiently approximated through beam search.

To use PHARAOH for tactical generation, we simply treat the source MRL as an NL, so that phrases in the MRL are sequences of consecutive MR symbols. Figure 5.2 illustrates the generation process. Note that the grammaticality of MRs is not an issue here, since they are given as input and are guaranteed to be grammatical.

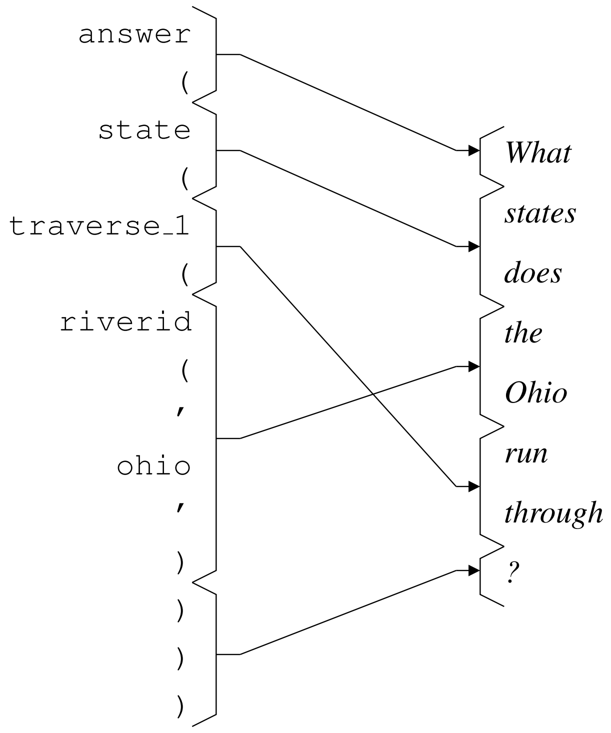


Figure 5.2: Generation using PHARAOH

5.2.2 WASP^{-1}: Generation by Inverting WASP

Tactical generation can be seen as the inverse of semantic parsing. In this section, we show how to invert the WASP semantic parsing algorithm to produce WASP^{-1}, and use it for tactical generation.

Recall that in WASP, the semantic parsing problem is formulated as a language translation task, where NL sentences are translated into formal MRs using an SCFG. Since an SCFG is fully symmetric with respect to both generated strings, it can also serve as the underlying formalism for generation. Figure 5.3 gives an overview of the WASP^{-1} algorithm. The shaded boxes show the components of the algorithm that are different from WASP (cf. Figure 3.3). Since WASP and WASP^{-1}

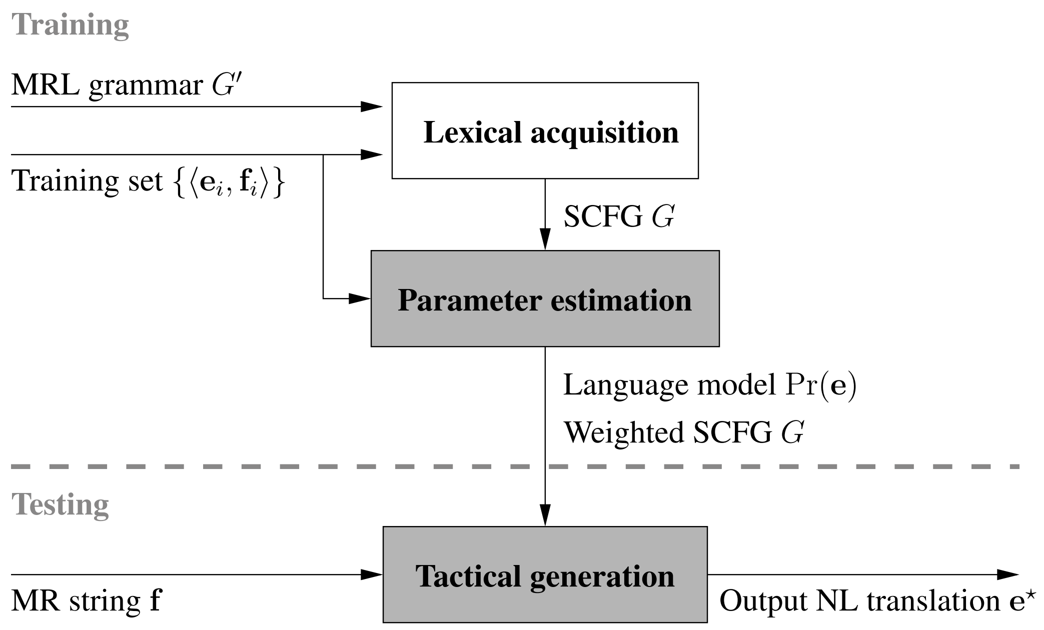


Figure 5.3: Overview of the WASP^{-1} tactical generation algorithm

share the same grammar, the lexical acquisition component is the same for both algorithms. However, as we will see shortly, the probabilistic model of WASP^{-1} is different from WASP, and as a result, WASP^{-1} uses a slightly different decoder.

Given an input MR, $\mathbf{f}$, WASP^{-1} finds a sentence $\mathbf{e}$ that maximizes the conditional probability $\Pr(\mathbf{e}|\mathbf{f})$. It is difficult to directly model $\Pr(\mathbf{e}|\mathbf{f})$, however, because it has to assign probabilities to output sentences that are not grammatical. There is no such requirement for semantic parsing with WASP, because the use of the MRL grammar ensures the grammaticality of all MRL translations. For generation, it is often hard to judge the grammaticality of an output sentence due to the inherent complexity of natural languages.

This motivates the noisy-channel framework for WASP^{-1}, where $\Pr(\mathbf{e}|\mathbf{f})$ is divided into two smaller components that are easier to model:

$$\mathbf{e}^\star = \arg\max_{\mathbf{e}} \Pr(\mathbf{e}|\mathbf{f}) = \arg\max_{\mathbf{e}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e}) \tag{5.3}$$

In this framework, $\Pr(\mathbf{e})$ is the *language model*, and $\Pr(\mathbf{f}|\mathbf{e})$ is the *translation model*. The generation task is to find an output NL translation, $\mathbf{e}^\star$, such that (1) it is a good sentence a priori, and (2) it preserves the meaning of the input MR. For the language model, we use an n-gram model, which has been found very useful in ranking candidate generated sentences (Knight and Hatzivassiloglou, 1995; Bangalore et al., 2000; Langkilde-Geary, 2002). For the translation model, we re-use the log-linear model of WASP (Equation 3.4). Hence computing $\mathbf{e}^\star$ means maximizing the following:

$$\begin{aligned} & \max_{\mathbf{e}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e}) \\ \approx\ & \max_{\mathbf{d} \in D(G|\mathbf{f})} \Pr(\mathbf{e}(\mathbf{d})) \Pr_\lambda(\mathbf{d}|\mathbf{e}(\mathbf{d})) \\ =\ & \max_{\mathbf{d} \in D(G|\mathbf{f})} \frac{\Pr(\mathbf{e}(\mathbf{d})) \cdot \exp \sum_i \lambda_i f_i(\mathbf{d})}{Z_\lambda(\mathbf{e}(\mathbf{d}))} \end{aligned} \tag{5.4}$$

where $D(G|\mathbf{f})$ is the set of all derivations that are consistent with $\mathbf{f}$ under an SCFG, G, and $\mathbf{e}(\mathbf{d})$ is the output sentence that a derivation $\mathbf{d}$ yields. The second line is due to the assumption that $\Pr(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{d} \in D(G|\mathbf{f})} \Pr(\mathbf{d}|\mathbf{e})$ is approximated by the Viterbi likelihood, $\max_{\mathbf{d} \in D(G|\mathbf{f})} \Pr(\mathbf{d}|\mathbf{e})$.

Learning under the noisy-channel framework thus involves two steps. First, a back-off n-gram language model with Good-Turing discounting and no lexical classes[1] is built from the training sentences using the SRILM toolkit (Stolcke, 2002). We use $n = 2$ since higher values seemed to cause overfitting in our experiments. Then a translation model is trained as described in Section 3.2.

[1]This is to ensure that the same language model is used in all systems that we tested.

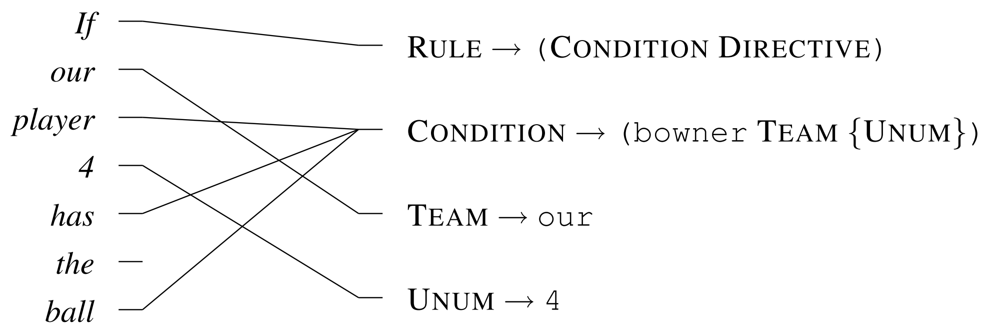

If
our
player
4
has
the
ball
RULE → (CONDITION DIRECTIVE)
CONDITION → (bowner TEAM {UNUM})
TEAM → our
UNUM → 4

Figure 5.4: A word alignment between English and CLANG (cf. Figure 3.5)

Compared to most existing work on generation, WASP^{-1} has the following characteristics:

1. It does not require any lexical information in the input MR, so lexical selection is an integral part of the decoding algorithm.

2. A lexical item may consist of multiple words. Moreover, it can be *discontiguous*.

The second characteristic is evident when we consider the following SCFG rule, which can be extracted from the word alignment in Figure 3.5, which is reproduced here in Figure 5.4 for convenience:

$$\textsc{Condition} \rightarrow \langle\ \textsc{Team}_{\boxed{1}}\ \textit{player}\ \textsc{Unum}_{\boxed{2}}\ \textit{has}\ (1)\ \textit{ball}\ ,\\ \texttt{(bowner}\ \textsc{Team}_{\boxed{1}}\ \texttt{\{}\textsc{Unum}_{\boxed{2}}\texttt{\})}\ \rangle$$

In this SCFG rule, the NL string contains a sequence of non-consecutive words, as in *our* **player** *4* **has the ball**. This lexical item is therefore discontiguous.

For decoding, we use an Earley chart generator that scans the input MR from left to right. This is possible because it is assumed that the order in which

MR symbols appear is fixed, i.e. the order determines the meaning of the MR.[2] Hence the chart generator is very similar to the chart parser in WASP, except for the following:

1. To facilitate the computation of the language model, chart items now include a list of $(n-1)$-grams that encode the context in which output NL phrases appear. The size of the list is $2N+2$, where N is the number of non-terminals to be rewritten in the partial derivation.

2. Words are generated from word gaps through special rules $(g) \rightarrow \langle \alpha, \emptyset \rangle$, where the word gap, (g), of size g is treated as a non-terminal, and α is the NL string that fills the gap ($|\alpha| \leq g$). The empty set symbol indicates that the gap filler does not carry any meaning. There are similar constructs in Carroll et al. (1999) for generating function words. Furthermore, to improve efficiency, the WASP^{-1} generator only considers gap fillers that have been observed during training.

3. The normalizing factor in (5.4), $Z_\lambda(\mathbf{e}(\mathbf{d}))$, is not a constant and varies across NL translations, $\mathbf{e}(\mathbf{d})$. (Note that $Z_\lambda(\mathbf{e})$ is constant for semantic parsing because $\mathbf{e}$ is given as input.) This is unfortunate because the calculation of $Z_\lambda(\mathbf{e}(\mathbf{d}))$ is expensive, and is not easy to incorporate into the chart generation algorithm. Decoding is thus performed through the following approximation: First, compute the k-best candidate NL translations based on the unnormalized version of (5.4), $\Pr(\mathbf{e}(\mathbf{d})) \cdot \exp \sum_i \lambda_i f_i(\mathbf{d})$. Then re-rank the list by normalizing the scores using $Z_\lambda(\mathbf{e}(\mathbf{d}))$, which is obtained by running

[2]See Chapter 6 where this assumption no longer holds.

the inside-outside algorithm on each NL translation. This results in a decoding algorithm that takes cubic time with respect to the length of *each* of the k candidate NL translations ($k = 100$ in our experiments).[3]

5.3 Improving the MT-based Generators

The MT-based generation algorithms, PHARAOH and WASP^{-1}, while reasonably effective, can be substantially improved by borrowing ideas from each other. This section describes the two resulting hybrid systems, PHARAOH++ (Section 5.3.1) and WASP^{-1}++ (Section 5.3.2).

5.3.1 Improving the PHARAOH-based Generator

A major weakness of PHARAOH as an NLG system is its inability to exploit the formal structure of the MRL. As with WASP^{-1}, the lexical acquisition algorithm of PHARAOH is based on the output of a word alignment model such as GIZA++, which performs poorly when applied directly to MRLs due to a large amount of semantically vacuous MR symbols (see Section 3.2.1).

We can improve the PHARAOH-based generator by supplying linearized MR parse trees as input rather than flat MR strings. As a result, the basic translation units are sequences of consecutive MRL productions in a linearized MR parse tree rather than sequences of consecutive symbols in an MR string. The same idea is used in WASP^{-1} to produce high-quality SCFG rules. We call the resulting hy-

[3]This k-best approximation can be avoided by choosing a formulation of $\Pr(\mathbf{e}|\mathbf{f})$ other than the noisy channel, e.g. $\Pr(\mathbf{e}(\mathbf{d}))\Pr_\lambda(\mathbf{d}|\mathbf{f})$. The latter probability can be computed using a log-linear model trained with an optimization criterion similar to Equation 3.6. Also Wu and Wong (1998) point out that normalization of the translation model may not be necessary when there is a strong language model. However, our experiments showed that normalization was necessary for WASP^{-1} to achieve good performance.

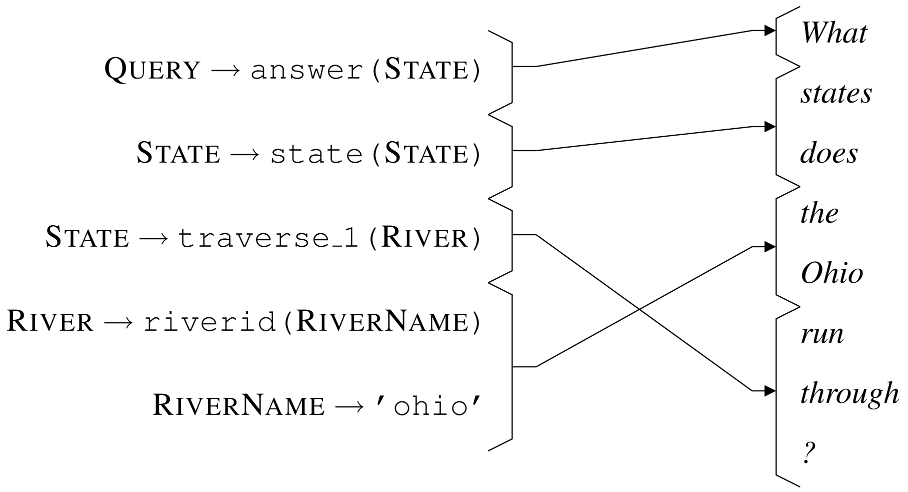


Figure 5.5: Generation using PHARAOH++

brid NLG system PHARAOH++. Figure 5.5 illustrates the generation process of PHARAOH++.

5.3.2 Improving the WASP^{-1} Algorithm

There are several aspects of PHARAOH that can be used to improve WASP^{-1}. First, the probabilistic model of WASP^{-1} is less than ideal as it requires an extra re-ranking step for normalization, which is expensive and prone to over-pruning. To remedy this situation, we can borrow the log-linear model of PHARAOH, and define the conditional probability of a derivation, $\mathbf{d}$, given an input MR string, $\mathbf{f}$, as:

$$\Pr_\lambda(\mathbf{d}|\mathbf{f}) \propto \Pr(\mathbf{e}(\mathbf{d}))^{\lambda_1} \prod_{d \in \mathbf{d}} w_\lambda(r(d)) \tag{5.5}$$

where $\prod_{d \in \mathbf{d}} w_\lambda(r(d))$ is the product of the weights of the SCFG rules used in a derivation $\mathbf{d}$. The weight w_λ of an SCFG rule is in turn defined as:

$$w_\lambda(A \to \langle \alpha, \beta \rangle) = P(\beta|\alpha)^{\lambda_2} P(\alpha|\beta)^{\lambda_3} P_w(\beta|\alpha)^{\lambda_4} P_w(\alpha|\beta)^{\lambda_5} \exp(-|\alpha|)^{\lambda_6} \tag{5.6}$$

where the relative frequencies, P, and lexical weights, P_w, are defined analogously to Equation 5.1. The word penalty, $\exp(-|\alpha|)$, offers a way to control the output sentence length. The output NL translation, $\mathbf{e}^\star$, is then the sentence that the most probable derivation consistent with $\mathbf{f}$ yields:

$$\mathbf{e}^\star = \mathbf{e}\left(\arg\max_{\mathbf{d}\in D(G|\mathbf{f})} \Pr_\lambda(\mathbf{d}|\mathbf{f})\right) \tag{5.7}$$

An advantage of this formulation of $\mathbf{e}^\star$ is that its computation requires no normalization and can be done exactly and efficiently. Also the model parameters λ are trained such that the BLEU score of the training set is directly maximized (Och, 2003). BLEU is a standard evaluation metric in the MT literature for assessing sentence fluency (Papineni et al., 2002).[4] Compared to the maximum conditional likelihood criterion used in WASP^{-1}, the maximum BLEU criterion is more strongly correlated with translation quality.

Following the phrase extraction algorithm in PHARAOH, we eliminate word gaps by incorporating unaligned words as part of the extracted NL strings. For example, given the word alignment in Figure 5.4, the following SCFG rule would be extracted instead of the one shown in Section 5.2.2, by incorporating the unaligned word *the* into the NL string:

$$\textsc{Condition} \rightarrow \langle\ \textsc{Team}_{\boxed{1}}\ \textit{player}\ \textsc{Unum}_{\boxed{2}}\ \textit{has the ball}\ ,\ \texttt{(bowner}\ \textsc{Team}_{\boxed{1}}\ \texttt{\{}\textsc{Unum}_{\boxed{2}}\texttt{\})}\ \rangle$$

The reason for eliminating word gaps is that while they are useful in dealing with unknown phrases during semantic parsing, for generation, using *known* phrases is generally preferred because it leads to better fluency. For a similar reason, we also allow the extraction of SCFG rules that are combinations of shorter SCFG rules.

[4]See Section 5.4.2 for a more detailed description of BLEU.

In other words, the extracted rules are not restricted to the shortest ones that cover the training set. This is because using *known combinations* of shorter phrases can lead to better fluency. For example, given the word alignment in Figure 5.4, rules would be extracted not only for individual MRL productions such as TEAM $\rightarrow$ `our` and UNUM $\rightarrow$ `4`, but also for combinations of productions such as CONDITION $\rightarrow$ `(bowner our {`UNUM`})` and RULE $\rightarrow$ `((bowner our {`UNUM`})` DIRECTIVE`)`. In this work, we restrict the number of productions being combined to be no more than 5.

The new hybrid system is called WASP^{-1}++. The main difference between PHARAOH++ and WASP^{-1}++ is that while PHARAOH++ only allows contiguous lexical items, WASP^{-1}++ also allows discontiguous lexical items. WASP^{-1}++ is also similar to the syntax-based MT system of Chiang (2005), which is based on an SCFG with PHARAOH's probabilistic model. The main differece is that we use the MRL grammar to constrain rule extraction, so that significantly fewer rules are extracted, leading to a learned grammar with much less ambiguity.

5.4 Experiments

This section describes the experiments that were performed to evaluate the four MT-based NLG systems that we introduced in this chapter, namely PHARAOH, WASP^{-1}, PHARAOH++, and WASP^{-1}++. We first present results from the automatic evauation (Section 5.4.2), followed by results from the human evaluation (Section 5.4.3). Then we show the experimental results on a multilingual data set (Section 5.4.4).

5.4.1 Data Sets

We evaluated the NLG systems in the GEOQUERY and ROBOCUP domains. The experimental results are based on the same corpora that were used for evaluating the WASP semantic parsing algorithm. In summary, the GEOQUERY corpus consists of 880 formal queries written in the functional query language FUNQL, along with their English translations. 250 of these queries were also annotated with Spanish, Japanese, and Turkish translations. The average sentence length for the 880-example English data set is 7.57 words. The ROBOCUP corpus consists of 300 pieces of coaching advice written in CLANG, along with their English translations. The average sentence length for the 300-example data set is 22.52 words. For the detailed corpus statistics, please refer to Table 3.1.

For each domain, there is a minimal set of lexical items representing knowledge needed for translating basic domain entities (Section 3.3.2). For GEOQUERY, the domain entities are various place names. For ROBOCUP, the domain entities are numbers and identifiers. Lexical items representing these domain entities are supplied to the MT-based generators as follows. For the PHARAOH-based generators, these lexical items are appended to the training set as separate sentence pairs, where each sentence pair corresponds to one domain entity. This method has been widely used in the statistical MT community for incorporating bilingual dictionaries as an additional knowledge source (Brown et al., 1993a; Och and Ney, 2000). For the WASP-based generators, these lexical items come in the form of SCFG rules, which are always included in the lexicon.

5.4.2 Automatic Evaluation

We performed 4 runs of standard 10-fold cross validation, and measured the performance of the learned generators using the BLEU score (Papineni et al.,

2002) and the NIST score (Doddington, 2002). Both automatic evaluation metrics approximate human assessment by comparing candidate translations with reference translations. Specifically, the BLEU score is the geometric mean of the precision of n-grams of various lengths, multiplied by a brevity penalty factor, BP, that penalizes candidate translations shorter than the reference translations:

$$\textsc{Bleu} = \text{BP} \cdot \exp \sum_{n=1}^{N} \frac{\log p_n}{N} \tag{5.8}$$

Here $N = 4$, and p_n denotes the n-gram precision of candidate translations (i.e. the proportion of n-grams that they share with the reference translations).[5] The NIST score is also based on n-gram co-occurrences, but it weighs more heavily those n-grams that occur less frequently (and hence are more informative). Also it uses an alternative brevity penalty factor, BP′, that minimizes the impact of *small* variations in the length of candidate translations (but penalizes large variations more heavily):

$$\textsc{Nist} = \text{BP}' \cdot \sum_{n=1}^{N} p'_n \tag{5.9}$$

Here $N = 5$, and p'_n denotes the weighted n-gram precision of candidate translations. BLEU and NIST are standard evaluation metrics in the MT literature (e.g. Koehn and Monz, 2006; NIST, 2006). Both of them have recently been used for evaluating NL generators (Langkilde-Geary, 2002; Nakanishi et al., 2005; Belz and Reiter, 2006).

[5]Each candidate translation may correspond to multiple reference translations, in which case the n-gram precision would increase. In the GEOQUERY corpus, some sentences are mapped to the same formal queries, so it is possible to supply multiple reference translations for each test example. We only used one reference translation per example, however, because n-to-1 mappings are relatively few, and the NIST MT evaluation script which we used only allows a constant number of reference translations for all test examples.

	GEOQUERY 880		ROBOCUP	
	BLEU	NIST	BLEU	NIST
PHARAOH	0.2070	3.1478	0.3247	5.0263
WASP^{-1}	0.4582	5.9900	0.4357	5.4486
PHARAOH++	**0.5354**	6.3637	0.4336	5.9185
WASP^{-1}++	**0.5370**	**6.4808**	**0.6022**	**6.8976**

Table 5.1: Automatic evaluation results for NL generators on the English corpora

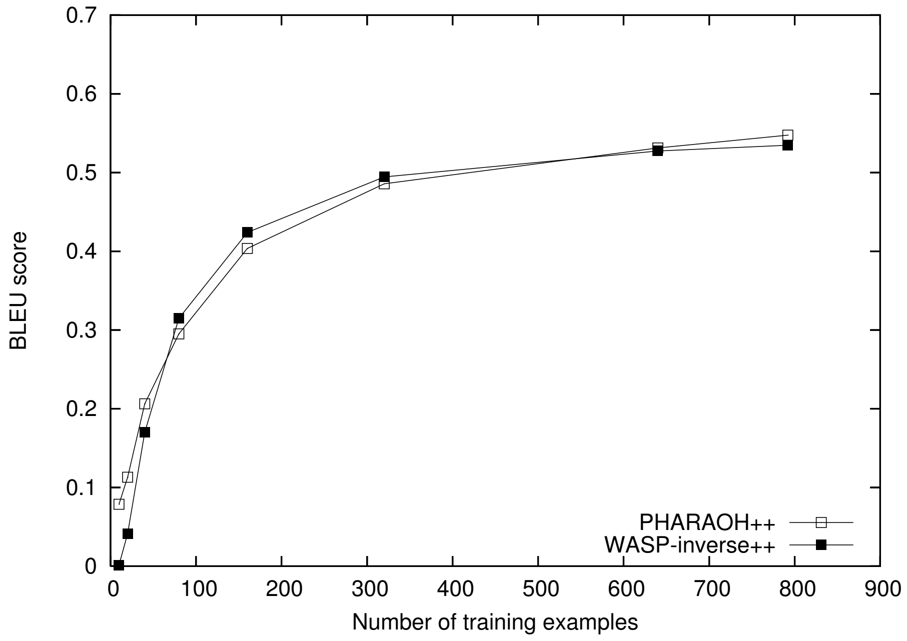
	GEOQUERY (880 data set)	ROBOCUP
PHARAOH	0.1 s	0.7 s
WASP^{-1}	2.4 s	49.7 s
PHARAOH++	0.03 s	0.1 s
WASP^{-1}++	0.7 s	8.2 s

Table 5.2: Average time needed for generating one test sentence

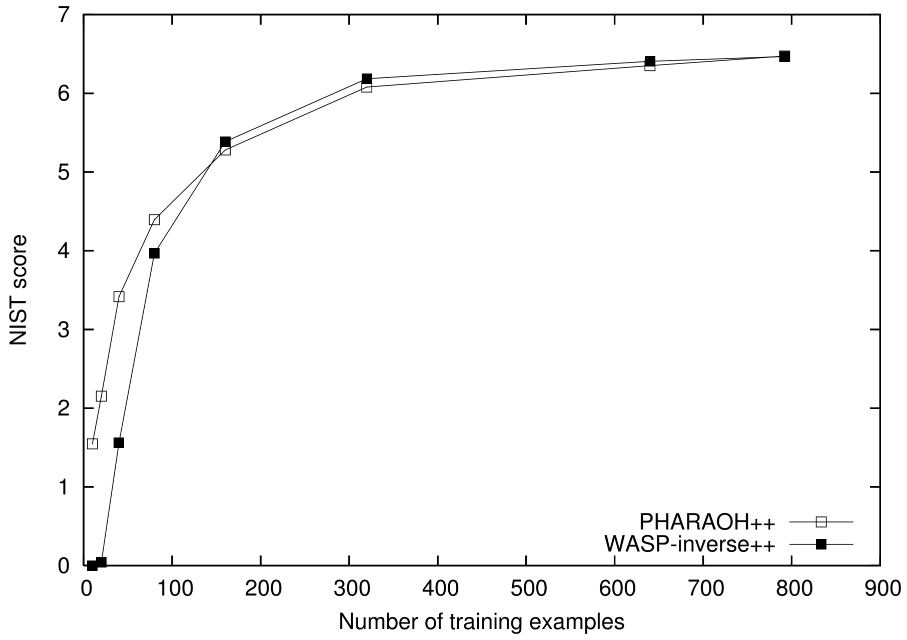
Table 5.1 presents the automatic evaluation results. The best-performing systems for each domain are shown in bold, where paired t-tests are used to determine statistical significance. Figures 5.6 and 5.7 show the BLEU and NIST learning curves for PHARAOH++ and WASP^{-1}++ (based on a *single* run of 10-fold cross validation).

A few observations can be made. First, WASP^{-1} produced more accurate NL generators than PHARAOH ($p < 0.001$). Second, PHARAOH++ significantly outperformed PHARAOH ($p < 0.001$). Both observations show the importance of exploiting the formal structure of the MRL. Third, WASP^{-1}++ significantly outperformed WASP^{-1} ($p < 0.001$). Much of the gain came from PHARAOH's probabilistic model. Decoding was also much faster (Table 5.2), despite exact inference and a larger grammar due to the extraction of longer SCFG rules.

Note that WASP^{-1}++ significantly outperformed PHARAOH++ with full train-

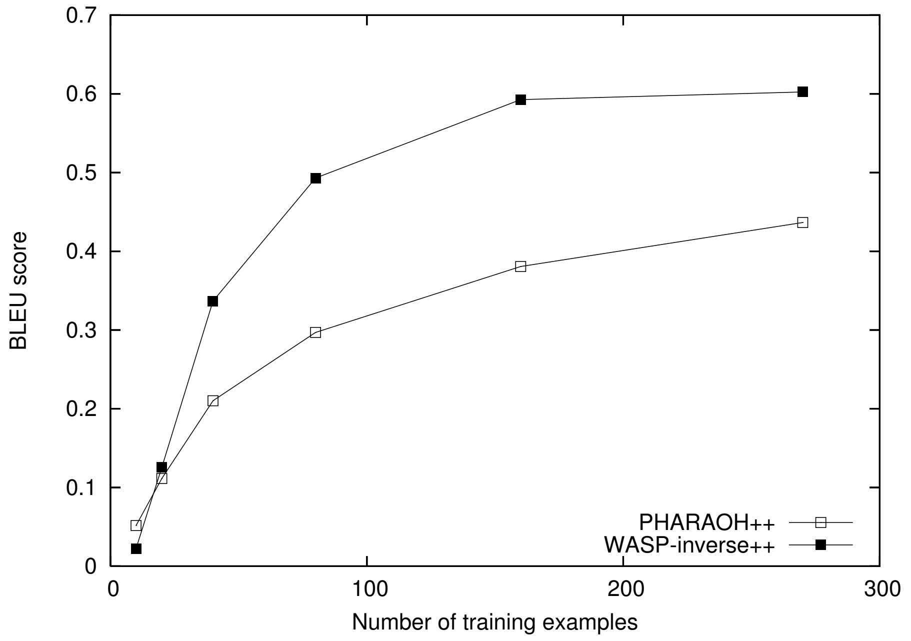


(a) BLEU score

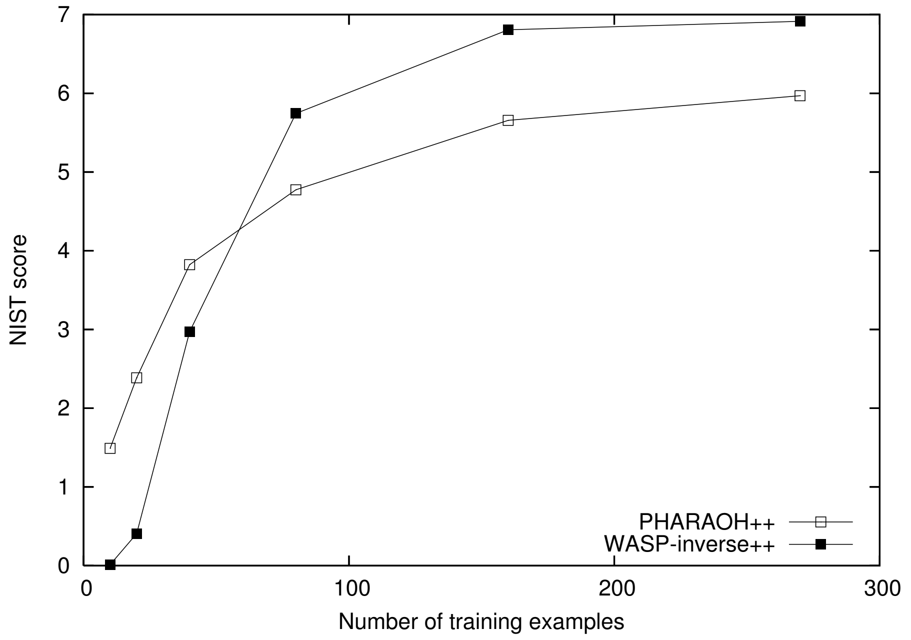


(b) NIST score

Figure 5.6: Learning curves for NL generators on the GEOQUERY 880 data set



(a) BLEU score



(b) NIST score

Figure 5.7: Learning curves for NL generators on the ROBOCUP data set

Reference: *If our player 2, 3, 7 or 5 has the ball and the ball is close to our goal line ...*

PHARAOH++: *If player 3 has the ball is in 2 5 the ball is in the area near our goal line ...*

WASP^{-1}++: *If players 2, 3, 7 and 5 has the ball and the ball is near our goal line ...*

Figure 5.8: Partial NL generator output in the ROBOCUP domain

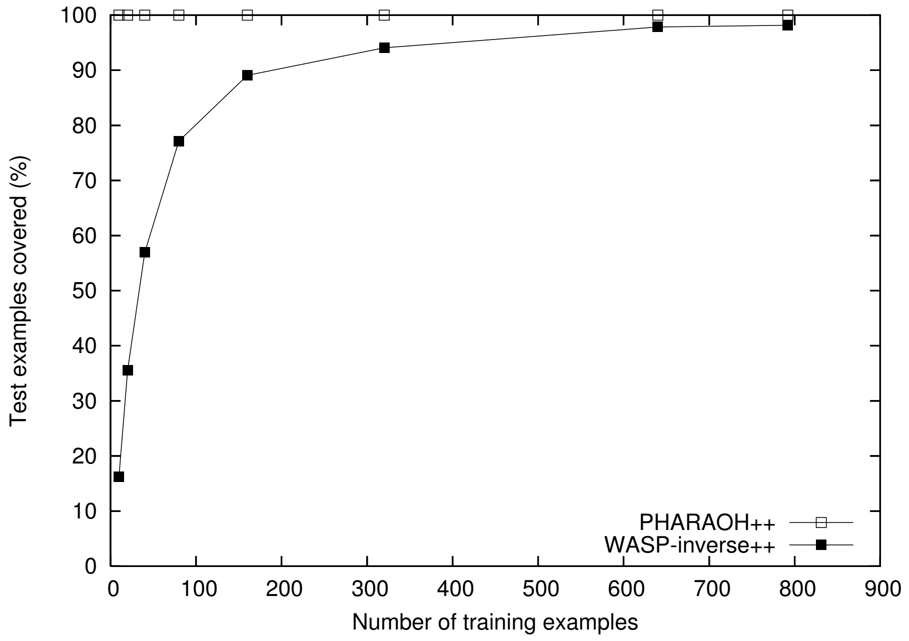
ing data in the ROBOCUP domain ($p < 0.001$). This is because WASP^{-1}++ allows discontiguous lexical items whereas PHARAOH++ does not. Such lexical items are commonly used in ROBOCUP for constructions like: **players** *2* **,** *3* **,** *7* **and** *5* (Figure 5.8); 26.96% of the lexical items that WASP^{-1}++ used during testing were discontiguous. When faced with such cases, PHARAOH++ would consistently omit some of the words (e.g. **players** *2 3 7 5*), or not learn any phrases for those constructions at all. As a result, given some input MRs, PHARAOH++ would fail to find fluent NL translations that preserve their meanings (Figure 5.8). On the other hand, for GEOQUERY, only 4.47% of the lexical items that WASP^{-1}++ used during testing were discontiguous, so the advantage of WASP^{-1}++ over PHARAOH++ was not as obvious ($p < 0.01$ for NIST, no significant difference for BLEU).

With limited training data, PHARAOH++ outperformed WASP^{-1}++ for both GEOQUERY and ROBOCUP domains (Figures 5.6 and 5.7). The reason is two-fold. First, PHARAOH++ learned simpler models than WASP^{-1}++ by restricting all lexical items to be contiguous. Second, PHARAOH++ had better coverage than WASP^{-1}++ given small training sets, i.e. more test examples received NL translations under PHARAOH++ than WASP^{-1}++ (Figure 5.9). This is because previously unseen MR predicates, left untranslated, are included in the output of PHARAOH++, ensuring 100% coverage. In contrast, WASP^{-1} would fail to produce any NL translations if there is any previously unseen predicate in an input MR, leading to high brevity penalty in the BLEU and NIST scores (especially for NIST). Note that although PHARAOH++ always generates some output, its output sentences, laden with MR symbols, are often unintelligible.

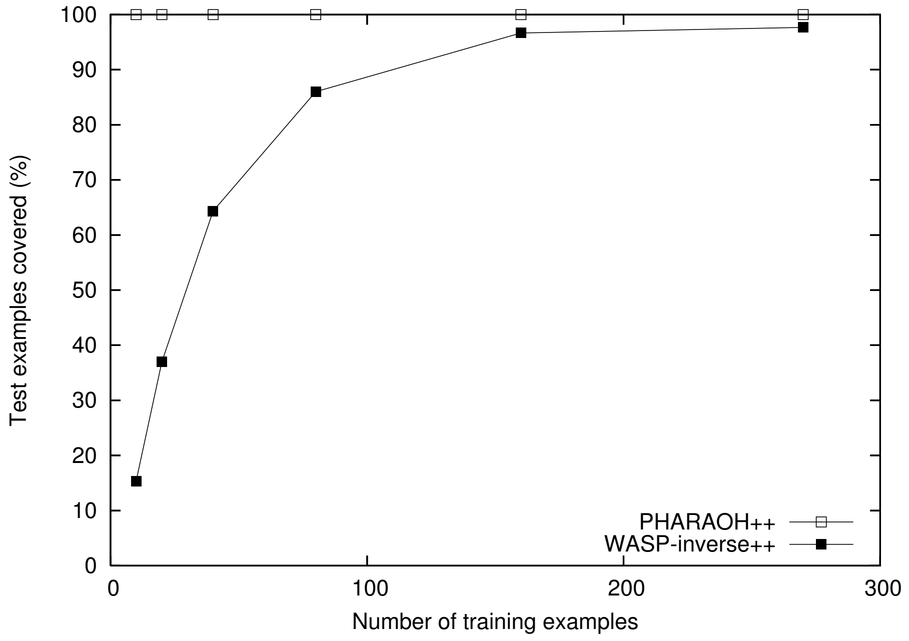
Our BLEU scores are not as high as those reported in Langkilde-Geary (2002) and Nakanishi et al. (2005), which are around 0.7 to 0.9. However, their work involves the re-generation of automatically parsed text, and the MRs that they use, which are essentially dependency parses, contain extensive lexical information of the target NL.

5.4.3 Human Evaluation

Automatic evaluation is only an imperfect substitute for human assessment. While it has been found that BLEU and NIST correlate quite well with human judgments in evaluating NLG systems (Belz and Reiter, 2006), it is best to support these figures with human evaluation, which did on a small scale. We recruited 4 native speakers of English with no previous experience with the GEOQUERY and ROBOCUP domains. Each subject was given the same 20 examples for each domain, randomly chosen from the test sets. For each example, the subjects were asked to judge the output of PHARAOH++ and WASP^{-1}++ in terms of *fluency* and *adequacy*. The fluency score shows how fluent a generated sentence is with no reference to what meaning it is supposed to convey. The adequacy score shows how well a generated sentence conveys the meaning of the reference sentence. The subjects were presented with the reference sentences in order to evaluate adequacy. They were also presented with the following definition of fluency and adequacy scores, adapted from Koehn and Monz (2006):



(a) GEOQUERY 880



(b) ROBOCUP

Figure 5.9: Coverage of NL generators on the English corpora

	GEOQUERY 880		ROBOCUP	
	Fluency	*Adequacy*	*Fluency*	*Adequacy*
PHARAOH++	**4.3**	**4.7**	2.5	2.9
WASP^{-1}++	**4.1**	**4.7**	**3.6**	**4.0**

Table 5.3: Human evaluation results for NL generators on the English corpora

Fluency score	*English proficiency*	*Adequacy score*	*Meaning conveyed*
5	Flawless English	5	All meaning
4	Good English	4	Most meaning
3	Non-native English	3	Some meaning
2	Disfluent English	2	Little meaning
1	Incomprehensible	1	No meaning

For each test example, we computed the average of the 4 human judges' scores. No score normalization was performed. Then we compared the two systems using paired t-tests. Table 5.3 shows that WASP^{-1}++ consistently produced good English sentences that preserved most of the meaning conveyed by the reference sentences. It also produced better NL generators than PHARAOH++ in the ROBOCUP domain ($p < 0.01$), which is consistent with the results of automatic evaluation.

5.4.4 Multilingual Experiments

Lastly, we describe our experiments on the multilingual GEOQUERY data set. Table 5.4 presents the automatic evaluation results for WASP^{-1}++ in four target NLs, namely English, Spanish, Japanese and Turkish, compared to PHARAOH++. Figure 5.10 shows the BLEU and NIST learning curves for WASP^{-1}++ (based on a *single* run of 10-fold cross validation). Similar to previous results on the larger GEOQUERY data set, WASP^{-1}++ outperformed PHARAOH++ for some language-metric pairs ($p < 0.05$), and otherwise performed comparably. Also consistent with

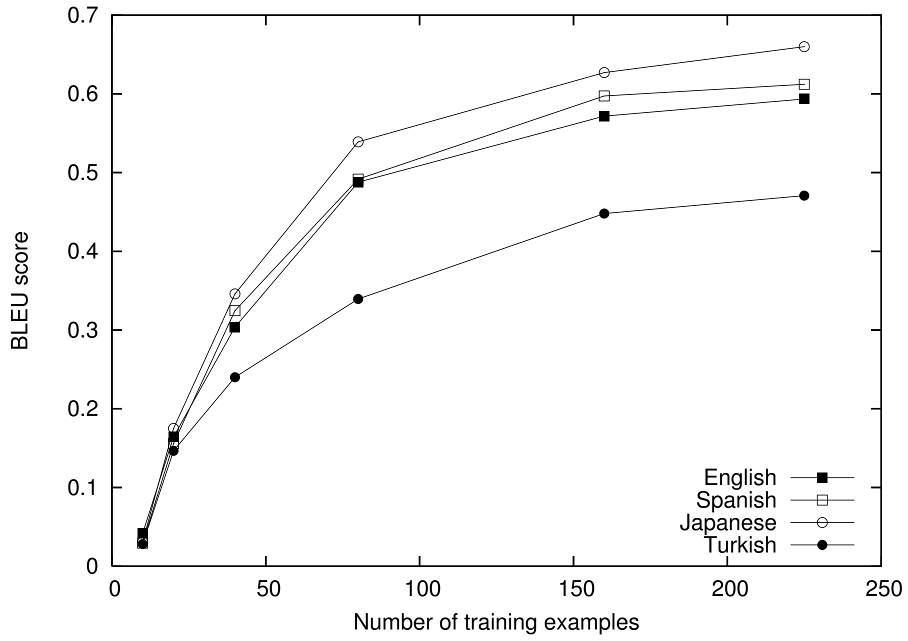
	PHARAOH++		WASP^{-1}++	
	BLEU	NIST	BLEU	NIST
English	0.5344	5.3289	**0.6035**	**5.7133**
Spanish	**0.6042**	**5.6321**	**0.6175**	**5.7293**
Japanese	0.6171	**4.5357**	**0.6585**	**4.6648**
Turkish	0.4562	**4.2220**	**0.4824**	**4.3283**

Table 5.4: Performance of WASP^{-1}++ on the multilingual GEOQUERY data set

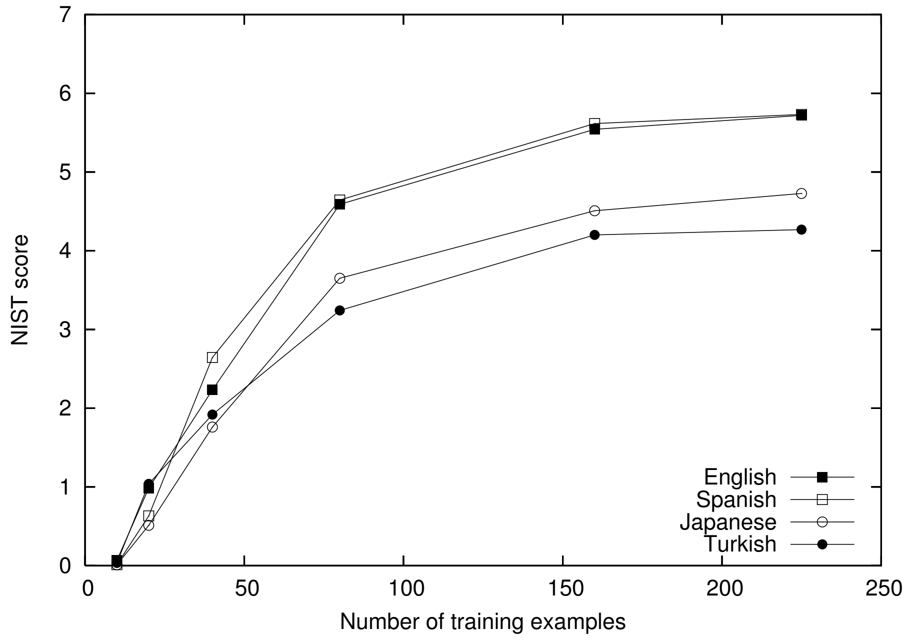
previous results for semantic parsing (Sections 3.3.3 and 4.3.2), the performance of the NLG systems was the lowest for Turkish, an agglutinative language with a relatively large vocabulary. The NIST scores for Japanese were also relatively low, although the BLEU scores were disproportionately high. A possible reason is that function morphemes, which are made separate tokens in the Japanese corpus, are given too much weight in the BLEU score.

5.5 Chapter Summary

In this chapter, we formulated the problem of tactical generation as a language translation task, where formal MRs are translated into NL sentences using statistical MT. We presented results on using a recent statistical MT system called PHAROAH for tactical generation. We also showed that the WASP semantic parsing algorithm can be inverted to produce a tactical generation system called WASP^{-1}. This approach allows the same learned grammar to be used for both parsing and generation. Also it allows the chart parser in WASP to be used for generation with minimal modifications. While reasonably effective, both PHARAOH and WASP^{-1} can be substantially improved by borrowing ideas from each other. Hence we presented two hybrid systems, PHARAOH++ and WASP^{-1}++. All four systems require source MRLs to be variable-free. We outlined a series of experiments that



(a) BLEU score



(b) NIST score

Figure 5.10: Learning curves for WASP^{-1}++ on multilingual GEOQUERY data

demonstrate the effectiveness of these tactical generation systems, based on automatic evaluation metrics and human assessment. The SCFG-based hybrid system WASP^{-1}++, produced by inverting WASP and incorporating PHARAOH's probabilistic model, was shown to achieve the best overall results across different languages and application domains.

Chapter 6

Natural Language Generation with Logical Forms

This chapter completes the last piece of the WASP puzzle, introducing a tactical generation algorithm that accepts input logical forms. The tactical generation algorithm, λ-WASP^{-1}++, is a straightforward extension of WASP^{-1}++, in which the underlying grammar is a λ-SCFG. This allows the same learned λ-SCFG to be used for both parsing and generation.

6.1 Motivation

As mentioned in Chapter 4, linguists have traditionally used predicate logic to represent meanings associated with NL expressions. Most existing NLG systems are based on predicate logic (White, 2004; Carroll and Oepen, 2005; Nakanishi et al., 2005). A prominent feature of predicate logic is its use of logical variables to denote entities. In Chapter 4, we showed how logical variables can be generated using a synchronous grammar, and how such a grammar can be learned from an annotated corpus for semantic parsing. An interesting problem would be to use the same learned grammar for NLG as well.

On the other hand, most, if not all, existing NLG systems that can handle input logical forms involve substantial human-engineered components that are difficult to maintain. For example, White (2004) describes a hybrid symbolic-statistical realizer under the OpenCCG framework, in which CCG grammars are hand-written. Carroll and Oepen (2005) describes a similar system using the English Resource Grammar (Copestake and Flickinger, 2000). Other NLG systems that are machine-learned typically require input representations that contain extensive lexical information of the target NL (Langkilde-Geary, 2002; Corston-Oliver et al., 2002; Nakanishi et al., 2005; Soricut and Marcu, 2006).

In this chapter, we show that the WASP^{-1}++ generation algorithm (Section 5.3.2) can be readily extended to support input logical forms.[1] The resulting algorithm, which we call λ-WASP^{-1}++, uses the same grammar as the λ-WASP semantic parser (Section 4.2). It automatically learns all of its linguistic knowledge from an annotated corpus consisting of NL sentences coupled with their correct logical forms. Moreover, it does not require any lexical information in the input representations, so lexical selection is an integral part of the decoding algorithm.

6.2 The λ-WASP^{-1}++ Algorithm

This section describes the λ-WASP^{-1}++ generation algorithm. We first give an overview of the chart generation algorithm (Section 6.2.1). Then we discuss k-best decoding for λ-WASP^{-1}++ (Section 6.2.2), which is needed for minimum error-rate training of the probabilistic model.

6.2.1 Overview

The λ-WASP^{-1}++ generation algorithm is a straightforward extension of WASP^{-1}++. Recall that in WASP^{-1}++, the problem of tactical generation is seen as translating formal MRs into NL sentences using an SCFG. WASP^{-1}++ uses a log-

[1]Although the WASP^{-1} algorithm (Section 5.2.2) can be modified in a similar way, we only consider WASP^{-1}++ because of its better performance.

linear model for parse disambiguation (Equation 5.5). An Earley chart generator that scans the input MR string from left to right is used for decoding, which is fine because it is assumed that the order in which MR symbols appear determines the meaning of the MR string.

In order to support input logical forms, we simply replace the underlying SCFG grammar of WASP^{-1}++ with a λ-SCFG (Section 4.2.1). The probabilistic model for generation remains unchanged. We call the resulting generation algorithm λ-WASP^{-1}++. To learn a λ-WASP^{-1}++ generator, we use the lexical acquisition algorithm described in Sections 4.2.2 and 4.2.4, and the parameter estimation algorithm described in Section 5.3.2.

However, there is a major difference between λ-WASP^{-1}++ and WASP^{-1}++. While in WASP^{-1}++, it can be safely assumed that the order in which MR symbols appear is significant, this assumption no longer holds in λ-WASP^{-1}. As mentioned in Section 4.2.4, certain logical operators such as conjunction (`,`) are associative and commutative. Hence, conjuncts can be reordered and regrouped without changing the meaning of a conjunction. In other words, the relative order of conjuncts in a conjunction is irrelevant. For example, given the following two input logical forms:

answer(x_1,(river(x_1),loc(x_1,x_2),
    equal(x_2,stateid(colorado))))

answer(x_1,(river(x_1),equal(x_2,stateid(colorado)),
    loc(x_1,x_2)))

(Name all the rivers in Colorado.)

the generated NL sentences should be identical, even though the relative order of conjuncts loc(x_1,x_2) and equal(x_2,stateid(colorado)) is different. This requires a different chart generator than the one used in WASP^{-1}++.

In this section, we describe a decoding algorithm that can handle input logical forms. As we will see in Section 6.2.2, this decoding algorithm is also used in minimum error-rate training of λ-WASP^{-1}++. Suppose that we have an λ-SCFG, G, which consists of the following rules:

r_1: QUERY $\rightarrow \langle$ *Name all the* FORM$_{\boxed{1}}$, `answer(`x_1`, (`FORM$_{\boxed{1}}$`(`x_1`)))` $\rangle$
r_2: FORM $\rightarrow \langle$ *rivers in* FORM$_{\boxed{1}}$,
λx_1`.river(`x_1`),loc(`x_1`,`x_2`),`FORM$_{\boxed{1}}$`(`x_2`)` $\rangle$
r_3: FORM $\rightarrow \langle$ STATE$_{\boxed{1}}$, λx_1`.equal(`x_1`,`STATE$_{\boxed{1}}$`)` $\rangle$
r_4: STATE $\rightarrow \langle$ STATENAME$_{\boxed{1}}$, `stateid(`STATENAME$_{\boxed{1}}$`)` $\rangle$
r_5: STATENAME $\rightarrow \langle$ *Colorado* , `colorado` $\rangle$

Given the following input logical form:

`answer(`x_1`,(river(`x_1`),equal(`x_2`,stateid(colorado)),`
`loc(`x_1`,`x_2`)))`

the decoding task is to find a derivation under G that is consistent with this logical form. Such a derivation exists, but only if we consider partial derivations that cover disjoint sets of input symbols. For example, the rule r_2 matches `river(`x_1`)` and `loc(`x_1`,`x_2`)` in the logical form, but these two formulas are separated by another formula `equal(`x_2`,stateid(colorado))`. Since a partial derivation may cover a disjoint set of input MR symbols, a chart item takes the form of a *coverage vector* with a bit for each formula (or term) in the input logical form showing whether the formula (or term) is covered by the chart item. The set of formulas (and terms) in a logical form can be found using the MRL grammar. For example, Figure 6.1 shows a parse tree for the logical form shown above. In this parse tree, each production correpsonds to a formula (e.g. `river(`x_1`)`) or a term that is not a logical variable (e.g. `colorado`). The relative order of these formulas and terms

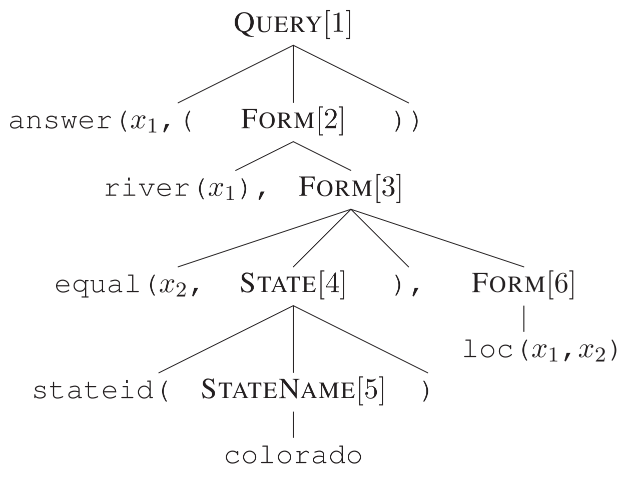


Figure 6.1: A parse tree for the sample Prolog logical form

are shown in bracketed indices, [1]–[6]. This ordering corresponds to the order of a top-down, left-most derivation. Each chart item for the sample logical form thus contains a coverage vector of 6 bits, a bit for each production in the parse tree. We use $[i, j, \ldots]$ to denote a bit vector in which bits $i, j, \ldots$ are set. The decoding algorithm starts with the creation of a set of initial chart items, which involves the computation of coverage vectors for each rule in G:

$$(r_1, [1], \{x_1/x_1\})$$
$$(r_2, [2, 6], \{x_1/x_1, x_2/x_2\})$$
$$(r_3, [3], \{x_1/x_2\})$$
$$(r_4, [4], \{\})$$
$$(r_5, [5], \{\})$$

Each chart item also contains a substitution, $\{x_1/x_{i_1}, \ldots, x_k/x_{i_k}\}$, that shows the renaming of logical variables necessary to transform the MR string of the rule into the part of the input logical form that the chart item covers. For example, for the rule r_3, the substitution is $\{x_1/x_2\}$, because the logical variable x_1 in the MR string of r_3 corresponds to x_2 in the input logical form. Note that each rule in G can give rise to multiple distinct chart items (or none at all). A chart item is said to be *inactive* if all RHS non-terminals in the rule have been rewritten. Otherwise, a chart item is said to be *active*. For example, all chart items shown above are active except the last one, as there are no RHS non-terminals in r_5.

Decoding proceeds by repeatedly combining chart items. An active item, (r_a, v_a, σ_a), may combine with an inactive item, (r_i, v_i, σ_i), if all of the following conditions are met:

1. The inactive item completes the active item.
2. The coverage vectors v_a and v_i are disjoint.
3. The substitution σ_i is compatible with σ_a.

To illustrate these conditions, consider the inactive item $(r_5, [5], \{\})$. It can combine with the active item $(r_4, [4], \{\})$, because [5] occupies the argument position of [4] (Condition 1), and [4] and [5] are disjoint (Condition 2). Condition 3 is also met because σ_i is empty. The combination of these two items results in a new item, $(r_4, [4–5], \{\})$, where [4–5] is the union of [4] and [5], and $\{\}$ is the union of σ_i and σ_a. This new item is inactive because all RHS non-terminals in r_4 have been rewritten.

This new item can then combine with the active item $(r_3, [3], \{x_1/x_2\})$, because [4–5] occupies the argument position of [3] (Condition 1), [3] and [4–5] are disjoint (Condition 2), and σ_i is empty (Condition 3). This results in a new inactive item, $(r_3, [3–5], \{x_1/x_2\})$, where $\{x_1/x_2\}$ is the union of σ_i and σ_a.

This new item can then combine with $(r_2, [2,6], \{x_1/x_1, x_2/x_2\})$: Condition 1 is met because $[2,6]$ and [3–5] together form a logical conjunction. Condition 2 is met because $[2,6]$ and [3–5] are disjoint. For Condition 3, note that the MR string of r_3, which is a λ-function, is used to rewrite the FORM non-terminal in the MR string of r_2. Upon function application, all bound occurrences of x_1 in the λ-function would be renamed to x_2, and therefore, occurrences of x_1 in σ_i should be renamed to x_2 as well. This results in a new substitution $\sigma'_i = \{x_2/x_2\}$, which is compatible with $\sigma_a = \{x_1/x_1, x_2/x_2\}$ because there is no x_j such that $x_j/x_{j'} \in \sigma'_i$, $x_j/x_{j''} \in \sigma_a$, and $x_{j'} \neq x_{j''}$. The combination of these two items thus gives rise to a new inactive item, $(r_2,$ [2–6]$, \{x_1/x_1, x_2/x_2\})$, where $\{x_1/x_1, x_2/x_2\}$ is the union of σ'_i and σ_a.

Lastly, this new item combines with $(r_1, [1], \{x_1/x_1\})$. The resulting item is $(r_1,$ [1–6]$, \{x_1/x_1\})$.[2] Since all 6 bits of the coverage vector are set, this item is a *goal item*, which corresponds to a complete derivation of the input logical form. The NL string that this derivation yields is then a translation of this logical form.

Figure 6.2 shows the basic decoding algorithm of λ-WASP^{-1}++. Inactive items are examined in ascending order of item size (i.e. number of true bits in the coverage vector). COMBINE-ITEMS(c, c') returns the item resulting from the combination of c and c'. It returns null if c and c' cannot combine. Each item is associated with a probability as defined by the log-linear model (Equation 5.5). UPDATE-CHART(C, c'') adds c'' to C if c'' is not already in C, or replaces the item in C with c'' if c'' has a higher probability. The output of this decoding algorithm is the most probable derivation consistent with the input logical form. This algorithm

[2]The substitution $\{x_1/x_1\}$ does not include any mapping from x_2, because x_2 is a free variable in r_2 and is no longer visible outside r_2. Following Kay (1996), we keep track of all logical variables that have become invisible (e.g. x_2). A partial derivation is filtered out if any of these logical variables is accidentally bound.

Input: a logical form, $\mathbf{f}$, a λ-SCFG, G, and an unambiguous MRL grammar, G'.

```
DECODE-$\lambda$-WASP$^{-1}$++$(\mathbf{f}, G, G')$
1  $\mathbf{f}' \leftarrow$ linearized parse of $\mathbf{f}$ under $G'$
2  $C \leftarrow$ set of initial chart items based on $\mathbf{f}'$ and $G$
3  for $i \leftarrow 1$ to $|\mathbf{f}'| - 1$
4      do for each inactive item $c \in C$ of size $i$
5             do for each active item $c' \in C$
6                    do $c'' \leftarrow$ COMBINE-ITEMS$(c, c')$
7                       if $c''$ is not null
8                         then UPDATE-CHART$(C, c'')$
9  return $c \in C$ of size $|\mathbf{f}'|$ with the highest probability
```

Figure 6.2: The basic decoding algorithm of λ-WASP^{-1}++

can take exponential time, since there can be $2^{|\mathbf{f}'|}$ distinct coverage vectors for a given logical form, $\mathbf{f}$. This seems reasonable because most other generation algorithms that accept input logical forms operate in exponential time as well (Moore, 2002; White, 2004; Carroll and Oepen, 2005). Moreover, generation can be sped up considerably by pruning away low-probability inactive items before each iteration of the outer **for** loop (i.e. before line 4). In our experiments, we retain only the top $100 \times |\mathbf{f}'|$ inactive items for each iteration.

6.2.2 k-Best Decoding

In λ-WASP^{-1}++, parameters of the probabilistic model are trained using minimum error-rate training, such that the BLEU score of the training set is directly maximized. Computation of BLEU requires actual generator output, and therefore, it involves decoding. Moreover, optimization of the BLEU score requires the computation of BLEU for *multiple* parameter settings. Och (2003) presents an efficient method for optimizing BLEU using log-linear models. The basic idea is to approximate the BLEU score by performing k-best decoding for only a handful of parameter settings.

In the previous section, we presented a 1-best decoding algorithm for λ-WASP^{-1}++. A naïve implementation of k-best decoding would compute the k-best derivations for every chart item. However, this can be prohibitively slow given that it already takes exponential time when $k = 1$. In this section, we describe an efficient k-best decoding algorithm for λ-WASP^{-1}++. Originally developed by Huang and Chiang (2005), this algorithm finds 100-best derivation lists almost as fast as 1-best decoding.

To see why the naïve implementation of k-best decoding is slow, consider the case where two chart items, c and c', combine to form a new chart item, c''. Finding the k-best derivations for c'' involves the following steps:

1. Enumerate k^2 derivations for c'', based on the k-best derivations for c and c''.
2. Sort these k^2 derivations.
3. Select the first k derivations from the sorted list of k^2 derivations.

This increases the time complexity of the decoder by a factor of $O(k^2 \log k)$. However, since we are only interested in the top k derivations for c'', the first two steps can be eliminated if we assume that:

1. The k-best lists for c and c' are sorted.
2. The function that computes the probability of a derivation is monotonic in each of its sub-derivations.[3]

[3]The use of a language model makes this function only approximately monotonic, e.g. certain combinations of common phrases can be highly unlikely. In this case, the k-best decoding algorithm is only approximate.

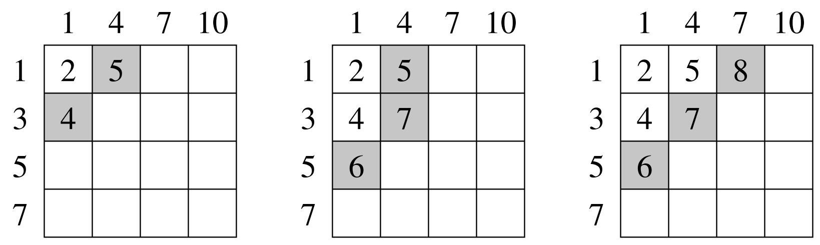


Figure 6.3: Example illustrating efficient k-best decoding

Let $c[i]$ be the i-th element in the k-best list for c. Given the assumptions above, it is clear that $c''[1]$ is the combination of $c[1]$ and $c'[1]$. Furthermore, $c''[2]$ is either the combination of $c[1]$ and $c'[2]$, or the combination of $c[2]$ and $c'[1]$. In general, if we view all possible combinations as a grid of cells (see Figure 6.3, where the numbers are negative log-probabilities), then the next cell to enumerate must be adjacent to the previously enumerated cells, i.e. it must be one of the cells shaded gray. Therefore, we need only consider $O(k)$ cells, and can safely ignore the rest of the grid.

From Figure 6.3, it is evident that to compute the k-best list for c'', we do not need the full k-best lists for c and c'. In general, since we are only interested in the k-best list for the goal items, we do not need the full k-best list for every item in the chart. As we go further down the derivation forest, the number of derivations required for each item becomes less and less. Therefore, we can speed up the k-best decoding algortihm considerably by computing k-best lists only when necessary. Details of the lazy computation of k-best lists can be found in Huang and Chiang (2005).

6.3 Experiments

In this section, we present experimental results that demonstrate the effectiveness of the λ-WASP^{-1}++ generation algorithm.

6.3.1 Data Sets and Methodology

We evaluated λ-WASP^{-1}++ in the GEOQUERY domain. In the experiments, we used the same GEOQUERY data set used to evaluate λ-WASP (Section 4.3.1). Specifically, the original Prolog logical forms were used. Table 4.1 shows the corpus statistics.

We only performed automatic evaluation, based on 4 runs of standard 10-fold cross validation, using the BLEU and NIST scores as the evaluation metrics. We did not perform human evaluation, since our human evaluation results in Section 5.4.3 indicate that the BLEU and NIST scores correlate well with human judgments in evaluating NLG systems in this domain.

6.3.2 Results and Discussion

Table 6.1 shows the performance of λ-WASP^{-1}++ on the GEOQUERY 880 data set with full training data, compared to two other NLG systems:

- PHARAOH++ (Section 5.3.1), which uses statistical phrase-based MT.
- WASP^{-1}++ (Section 5.3.2), the inverse of the WASP semantic parser, with PHARAOH's probabilistic model.

Unlike λ-WASP^{-1}++, both PHARAOH++ and WASP^{-1}++ take functional queries (written in FUNQL) as input. The best-performing systems based on paired t-tests

	GEOQUERY (880 data set)	
	BLEU	NIST
λ-WASP^{-1}++	**0.5320**	**6.4668**
PHARAOH++	**0.5354**	6.3637
WASP^{-1}++	**0.5370**	**6.4808**

Table 6.1: Performance of λ-WASP^{-1}++ on the GEOQUERY 880 data set

	GEOQUERY (880 data set)
λ-WASP^{-1}++	2.9 s
PHARAOH++	0.03 s
WASP^{-1}++	0.7 s

Table 6.2: Average time needed for generating one test sentence

are shown in bold ($p < 0.05$). Figure 6.4 shows the learning curves (based on a *single* run of 10-fold cross validation).

Table 6.1 shows that the performance of λ-WASP^{-1}++ is comparable to that of PHARAOH++ and WASP^{-1}++, despite markedly different input representations. Pruning also kept the running time to a reasonable level (Table 6.2), although the decoding algorithm could take exponential time.

Figure 6.4 shows that λ-WASP^{-1}++ outperformed WASP^{-1}++ with limited training data. This is because the lexical acquisition algorithm of λ-WASP^{-1}++ (i.e. that of λ-WASP) produces rules that generalize better (Section 4.2.4). Hence coverage is significantly higher for λ-WASP^{-1}++, especially when the training set is small (Figure 6.5), leading to steeper learning curves in terms of the BLEU and NIST scores. However, WASP^{-1}++ quickly caught up in terms of coverage as more training data was available. This is unlike the parsing case where WASP failed to keep up with λ-WASP in terms of recall (Figure 4.11). This indicates that tactical generation is an easier task than semantic parsing. While for tactical generation

	PHARAOH++		WASP^{-1}++		λ-WASP^{-1}++	
	BLEU	NIST	BLEU	NIST	BLEU	NIST
English	0.5344	5.3289	**0.6035**	**5.7133**	**0.6121**	**5.8254**
Spanish	0.6042	5.6321	0.6175	5.7293	**0.6584**	**5.9390**
Japanese	0.6171	4.5357	**0.6585**	**4.6648**	**0.6857**	**4.8330**
Turkish	0.4562	4.2220	**0.4824**	**4.3283**	**0.4737**	**4.3553**

Table 6.3: Performance of λ-WASP^{-1}++ on multilingual GEOQUERY data

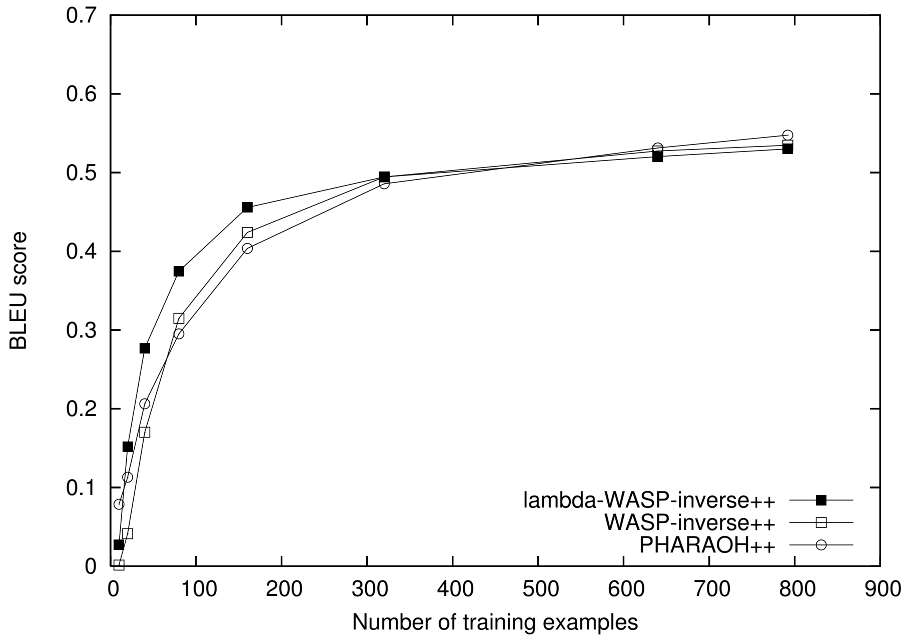
it suffices to learn one mapping for each MR predicate to get complete coverage, for semantic parsing one needs to learn a mapping for each *NL phrase* to achieve perfect recall, which is much more difficult because of synonymy.

Besides, λ-WASP^{-1}++ outperformed PHARAOH++ when the training set was small. This indicates that despite its lower coverage compared to PHARAOH++, λ-WASP^{-1}++ produced NL translations that were consistently more accurate.

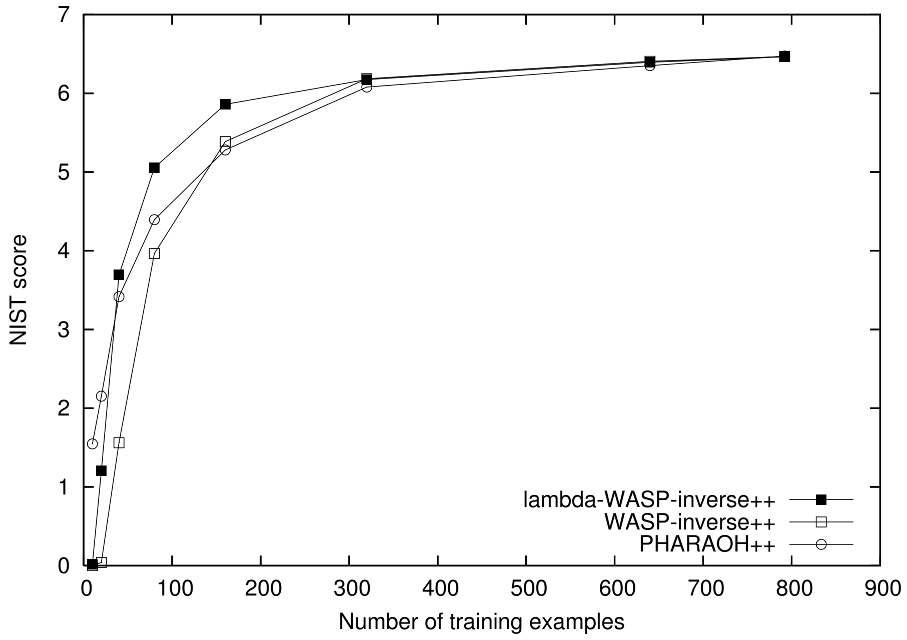
Table 6.3 and Figure 6.6 show the performance of λ-WASP^{-1}++ on the multilingual GEOQUERY data set. Similar to previous results on the larger GEOQUERY data set, λ-WASP^{-1}++ outperformed PHARAOH++, and performed comparably to WASP^{-1}++. Also consistent with previous observations (Section 5.4.4), the performance of λ-WASP^{-1}++ is the lowest for Turkish, followed by Japanese. For English and Spanish, the performance is comparable.

6.4 Chapter Summary

In this chapter, we described a tactical generation algorithm that translates logical forms into NL sentences. This algorithm is called λ-WASP^{-1}++. It can be seen as the inverse of the λ-WASP semantic parser, since both algorithms are based on the same underlying λ-SCFG grammar. It also shares the same log-linear proba-



(a) BLEU score



(b) NIST score

Figure 6.4: Learning curves for λ-WASP^{-1}++ on the GEOQUERY 880 data set

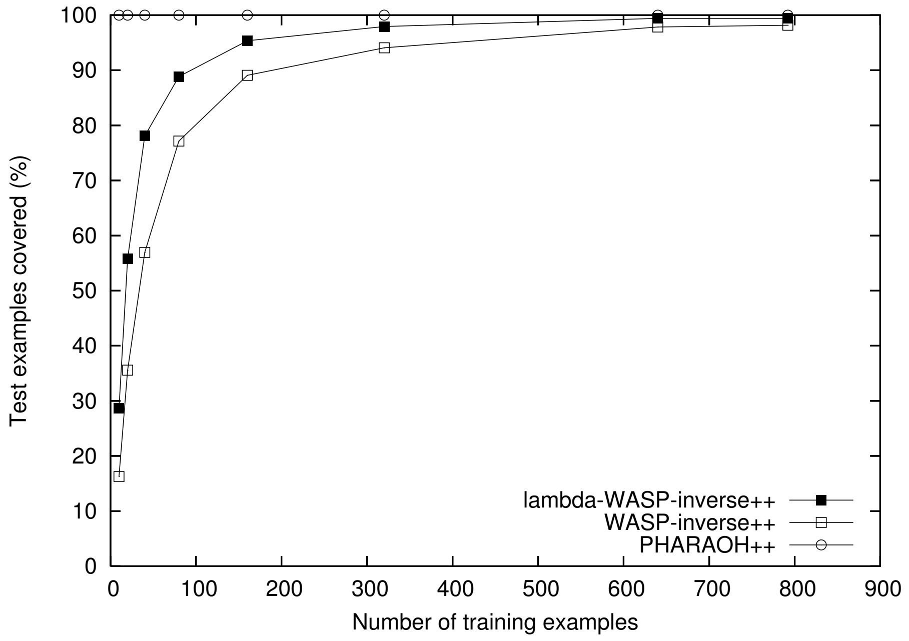
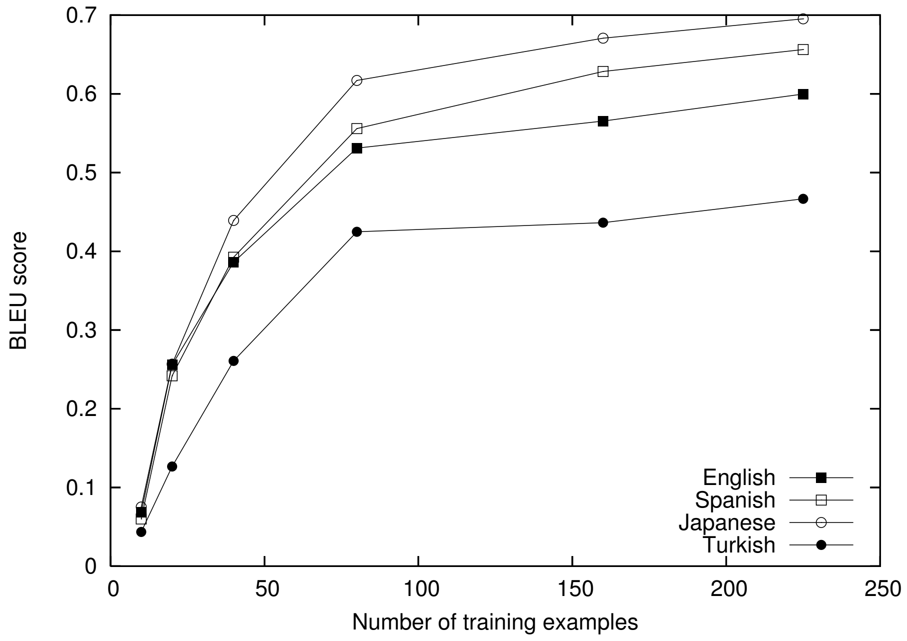
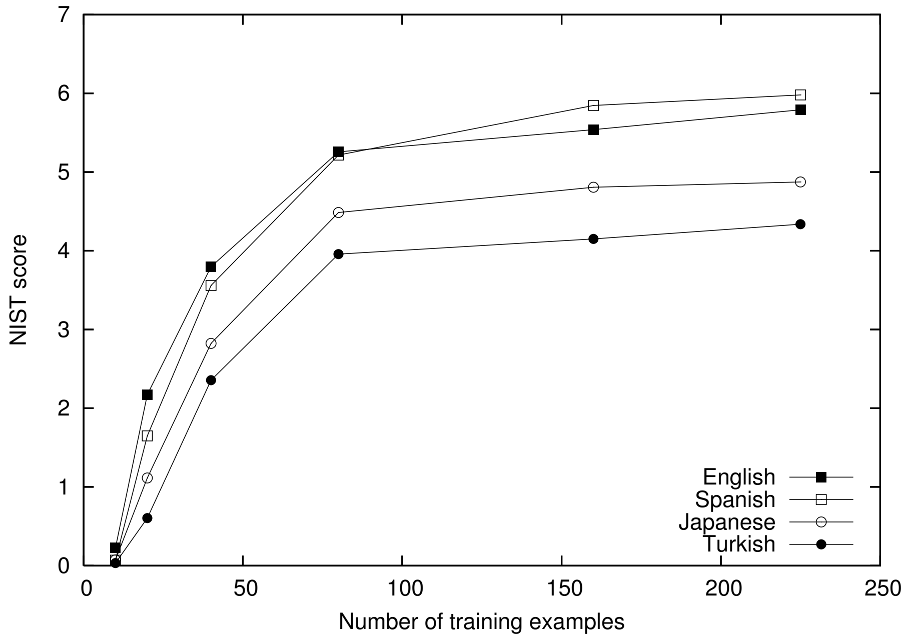


Figure 6.5: Coverage of λ-WASP^{-1}++ on the GEOQUERY 880 data set

bilistic model with WASP^{-1}++, which is maximum-BLEU trained. We described a chart generation algorithm that can handle input logical forms, and a fast k-best decoding algorithm for efficient maximum-BLEU training. Experiments showed that λ-WASP^{-1}++ is competitive compared to other MT-based generators, especially when training data is scarce.



(a) BLEU score



(b) NIST score

Figure 6.6: Learning curves for λ-WASP^{-1}++ on multilingual GEOQUERY data

Chapter 7

Future Work

In this chapter, we discuss some future directions for the research presented in this thesis.

7.1 Interlingual Machine Translation

As mentioned in Chapter 1, an application of semantic parsers and tactical generators is *interlingual MT*. In interlingual MT, source texts are first converted into a formal MRL that is language-independent, called an *interlingua*. From such interlingual representations, target texts are then generated. A possible interlingua in the GEOQUERY domain, for example, would be Prolog logical forms. An advantage of interlingual MT over direct MT is economy of effort in a multilingual environment: While direct MT requires a separate system for each language *pair*, interlingual MT only requires a parser and a generator for each language. Moreover, for structurally dissimilar language pairs such as Turkish and English, interlingual MT can achieve good results with a simpler system design (Hakkani et al., 1998). Early knowledge-based, interlingual MT systems are effective in restricted domains with limited vocabulary (Nyberg and Mitamura, 1992). It would be interesting to see how statistical interlingual MT systems compare against state-of-the-art direct MT systems (e.g. PHARAOH) in restricted domains such as GEOQUERY.

We evaluated a simple statistical interlingual MT system composed of λ-

	λ-WASP/λ-WASP^{-1}++			PHARAOH		
	Cov. (%)	BLEU	NIST	*Cov.* (%)	BLEU	NIST
Spanish-English	90.4	0.5415	5.1790	**100.0**	**0.7496**	**6.6862**
Japanese-English	90.0	0.5255	**5.2691**	**100.0**	**0.5700**	**5.6039**
Turkish-English	77.6	0.4431	3.8735	**100.0**	**0.6490**	**6.1504**

Table 7.1: Performance of MT systems on multilingual GEOQUERY data

	λ-WASP/λ-WASP^{-1}++		PHARAOH	
	BLEU	NIST	BLEU	NIST
Spanish-English	0.6215	5.8076	**0.7836**	**6.6443**
Japanese-English	**0.5930**	**5.6748**	**0.6149**	**5.7997**
Turkish-English	0.6218	5.6258	**0.7503**	**6.4653**

Table 7.2: MT performance considering only examples covered by both systems

WASP (Section 4.2) and λ-WASP^{-1}++ (Section 6.2). In this MT system, source sentences are converted into Prolog logical forms using λ-WASP. Then, the Prolog logical forms are translated into the target language using λ-WASP^{-1}++. For each source sentence, only the best Prolog logical form is used. If the source sentence cannot be converted into a complete Prolog logical form, then no output sentence will be generated.

Table 7.1 shows the preliminary results on the multilingual GEOQUERY data set using 10-fold cross validation, where the best-performing systems based on paired t-tests are shown in bold. Besides the BLEU and NIST scores, the table also shows the percentage of test examples covered by the MT systems. By most measures, PHARAOH outperformed the interlingual MT system ($p < 0.05$ based on paired t-tests). A primary reason is that λ-WASP often could not analyze source sentences completely, which led to low coverage. However, even ignoring sentences that are not covered, the performance of the interlingual MT system is

Source: *¿Cuántas personas viven en Spokane, Washington?*
Reference: *How many people live in Spokane, Washington?*
λ-WASP/λ-WASP^{-1}++: *What is the population of Spokane, WA?*

Figure 7.1: Output of interlingual MT from Spanish to English in GEOQUERY

still low (Table 7.2). There are two contributing factors to this. First, in the interlingual MT system, the parsing and generation components are loosely coupled, so error easily propagates. Second, the interlingua may fail to capture certain stylistic preferences in the texts.

Some of these problems could be easily remedied. To improve coverage, we could add rules to λ-WASP and λ-WASP^{-1}++ that glue partial derivations together (Chiang, 2005), or add default rules for previously unseen words. To reduce error propagation, we could have λ-WASP produce multiple analyses of a source sentence to avoid committing to a particular analysis. λ-WASP^{-1}++ could then be used to generate the best overall translation, or a translation that covers the most analyses (Knight and Langkilde, 2000). To make sure that synonymous expressions do not get penalized (e.g. Figure 7.1), we could elicit more reference translations for each source sentence (Section 5.4.2), or perform human evaluation (Section 5.4.3).

A more fundamental problem is designing an appropriate interlingua for a particular domain. An MRL that is adequate for querying databases may not be adequate for interlingual MT. Moreover, while it is feasible to build an interlingual MT system for specific domains such as medical triage (Gao et al., 2006), it is much more difficult for broader domains such as newspaper texts (Knight et al., 1995; Farwell et al., 2004). This is because to describe all important concepts in the world requires a comprehensive ontology, but such knowledge resources are very difficult to obtain. However, we still believe that translation involves understanding, and interlingual MT is the right approach. As we mentioned in Chapter 1, the use of concise interlingual representations can improve statistical MT. Likewise, the ability to understand unrestricted texts will have wide implications in other research areas such as question answering, information retrieval, document summarization, and human-computer interaction. In subsequent sections, we will discuss some possible research avenues that would allow progress toward broad-domain natural language understanding and generation.

7.2 Shallow Semantic Parsing

Current research on broad-domain semantic analysis has mainly focused on the following two sub-tasks: *word sense disambiguation* and *semantic role labeling*. Word sense disambiguation (WSD) is to identify the correct meaning (or *sense*) of a word in context (Lee and Ng, 2002). Semantic role labeling (SRL) is to identify the semantic arguments of a given predicate in a sentence (Gildea and Jurafsky, 2002). These two tasks are closely related. For example, consider the following sentence:

The robbers tied Peter to his chair.

To identify the predicate-argument structure of this sentence, we need to determine the correct sense of the word *tied*, which is "to physically attach" in this case (as opposed to "making a mental connection"). Once the predicate is correctly identified, we can identify its arguments (shown in brackets):

[$_{\text{AGENT}}$ *The robbers*] **tied** [$_{\text{ITEM}}$ *Peter*] [$_{\text{GOAL}}$ *to his chair*] .

Each argument takes a specific *role*. In this case, the robbers are the AGENT that causes Peter (the ITEM) to be physically attached to his chair (the GOAL). These roles can be predicate-specific. In other words, the sense of a word (e.g. *tied*) can influence the roles associated with it. Conversely, the roles associated with a word can influence its sense as well (Lapata and Brew, 1999). In this case, the fact that a chair is a physical GOAL makes it more likely that *tied* means "to physically attach". These word senses and semantic roles are defined in ontologies such as WordNet (Fellbaum, 1998), FrameNet (Fillmore et al., 2003), and Omega (Philpot et al., 2005).

Traditionally, WSD and SRL have been treated as two separate tasks: WSD is done without knowledge of the semantic roles associated with a word, and SRL is done assuming that the predicate has been correctly identified (Gildea and Jurafsky, 2002), or assuming that the semantic roles are predicate-independent as in PropBank (Palmer et al., 2005). We argue that WSD and SRL should be more tightly coupled. The need for joint inference is more evident when we consider more than one predicate in a sentence:

> [$_{\text{RECIPIENT}}$ *Mary*] **got** [$_{\text{THEME}}$ [$_{\text{REQUIREMENT}}$ *the ingredients*] **needed** [$_{\text{DEPENDENT}}$ *to* **make** [$_{\text{FOOD}}$ *ice-cream*]]] .

In this sentence, the predicates and their arguments form a tree structure. However, current SRL methods that consider one predicate at a time cannot capture such interactions among predicates (Carreras and Màrquez, 2005; Erk and Padó, 2006).

There has been some preliminary work on combining WSD with SRL. Thompson et al. (2003) present a generative model that performs joint WSD and SRL for the main verb of a sentence. Erk (2005) reports some preliminary results on using semantic argument information of a word to improve WSD.

In the future, we would like to explore semantic parsing in a broad-domain setting. Specifically, we would like to combine WSD with SRL in a more tightly-coupled process. The semantic parsing task is *shallow* in the sense that many important linguistic phenomena are ignored, such as quantification and tense. Also words can be left unanalyzed if they do not correspond to any defined concepts in an ontology. Note that in WASP, WSD and SRL are already integrated in the chart parsing process. We believe that WASP could be adapted to handle unrestricted texts, by treating nested predicate-argument structures as the target MRL.

The following semantically-annotated corpora could be used for broad-domain shallow semantic parsing. Baker et al. (2007) have recently released a small English corpus based on FrameNet, in which every sentence is annotated with the semantic arguments for all predicates. For larger corpora, the OntoNotes project (Weischedel et al., 2007) is an ongoing effort to produce an extended version of English, Chinese and Arabic Propbanks annotated with word sense information for nouns and verbs, linked to the Omega ontology, and coreference.

7.3 Beyond Context-Free Grammars

Another issue related to broad-domain semantic analysis is the prevalence of long-distance dependencies in unrestricted texts.[1] Long-distance dependencies occur when semantic arguments are realized outside the maximal phrase headed by the predicate. Examples include the following:

[*The dog*] *which they had just* **bought** *ran away*. (Relative clause)

[*They*] *are hoping to* **secure** *state funding this year*. (Subject control)

[*This record*] *is hard to* **beat**. (*Tough*-movement)

[1]Long-distance dependencies are not very common in the restricted domains we have worked with. For example, λ-WASP outperforms Zettlemoyer and Collins (2007) in the GEOQUERY domain (Section 4.3.2), although the latter can handle long-distance dependencies.

It is well known that CFGs cannot easily capture long-distance dependencies (Levy and Manning, 2004). A number of sophisticated grammar formalisms that can handle such dependencies have been developed, including combinatory categorial grammars (CCG) and tree-adjoining grammars (TAG). CCGs and TAGs are also said to be *mildly context-sensitive*, because they have strictly greater generative capacity than CFGs, yet remain polynomially parsable (Weir, 1988). Recently, Clark and Curran (2004) released a highly efficient wide-coverage CCG parser, which provides an attractive alternative to traditional statistical CFG parsers (Collins, 1997; Charniak, 2000).

Existing work on semantic analysis using CCGs and TAGs mostly involves hand-written components that are language-specific (Shieber and Schabes, 1990b; Bos, 2005; Zettlemoyer and Collins, 2007). In the future, we would like to devise learning algorithms similar to WASP that construct synchronous CCGs and TAGs given training data in any language. Such synchronous grammars can be useful in natural language generation and machine translation as well (Shieber and Schabes, 1990a; Shieber, 2007). Our goal is to extract synchronous grammars from parallel corpora with limited or no syntactic annotations. For this, previous work on extracting CCGs and TAGs from non-CCG or TAG-annotated corpora would be relevant (Hockenmaier and Steedman, 2002; Chen et al., 2006).

7.4 Using Ontologies in Semantic Parsing

The research presented in this thesis illustrates the importance of domain knowledge in semantic parsing and natural language generation. Specifically, in all of the WASP-based systems, domain knowledge comes in the form of an MRL grammar that defines a set of possible MRs in a particular domain. However, not all information can be conveniently encoded in an MRL grammar, and for broad-domain semantic analysis, knowledge bases such as FrameNet and Omega can be very useful. An interesting question would be how to effectively use the knowledge encoded in these ontologies in a statistical semantic parsing framework.

On the other hand, knowledge gleaned from texts can also be integrated with existing ontologies, which can be useful for understanding further texts in the same domain (Barker et al., 2007). In other words, natural language understanding and knowledge acquisition can form a tightly-coupled cycle, where knowledge is accumulated by reading a given corpus of *unannotated* texts. This would allow knowledge acquisition on a truly large scale, and can lead to automated systems that learn natural languages like humans, using basic prior knowledge to bootstrap the learning process. To combine natural language understanding and knowledge acquisition in a robust statistical framework is therefore a very interesting problem, which we intend to pursue in the future.

Chapter 8

Conclusions

In this thesis, we focused on two sub-tasks of natural language understanding and generation, namely semantic parsing and tactical generation. Semantic parsing is the task of transforming natural-language sentences into formal symbolic meaning representations (MR), and tactical generation is the inverse task of transforming formal MRs into sentences. We presented a number of novel statistical learning algorithms for semantic parsing and tactical generation. These algorithms automatically learn all of their linguistic knowledge from annotated corpora, and can handle sentences that are conceptually complex.

The key idea of this thesis is that since both semantic parsing and tactical generation are essentially language translation tasks between natural languages (NL) and formal meaning representation languages (MRL), both can be tackled using state-of-the-art statistical machine translation (MT) techniques. Specifically, we introduced a learning algorithm for semantic parsing called WASP (Chapter 3), based on a technique called synchronous parsing, which has been extensively used in syntax-based statistical MT. The underlying grammar of WASP is a weighted synchronous context-free grammar (SCFG) extracted from an automatically word-aligned parallel corpus consisting of NL sentences and their correct MRs, with the help of an unambiguous context-free grammar of the target MRL. The weights of the SCFG define a log-linear distribution over its derivations. The WASP algorithm is designed to handle variable-free MRLs, as exemplified by CLANG, the

ROBOCUP coach language (Section 2.1). We empirically evaluated the effectiveness of WASP in two real-world domains, GEOQUERY and ROBOCUP, and in four different NLs, namely English, Spanish, Japanese and Turkish. Experimental results showed that the performance of WASP is competitive compared to the currently best methods requiring similar supervision.

In Chapter 4, we extended the WASP semantic parsing algorithm to handle MRLs such as predicate logic, on which most existing work on formal semantics and computational semantics is based. The resulting algorithm, λ-WASP, uses an extended version of SCFG called λ-SCFG, in which logical forms are generated using the lambda calculus. We proposed a learning algorithm similar to WASP, which learns a weighted λ-SCFG from a parallel corpus consisting of NL sentences paired with their correct logical forms. We further refined the learning algorithm through transformation of logical forms and language modeling for target MRLs. Using the same amount of supervision, λ-WASP was shown to significantly outperform WASP, and is currently one of the best semantic parsing algorithms in the GEOQUERY domain.

For tactical generation, we proposed several learning methods for variable-free MRLs using statistical MT (Chapter 5). We presented results on using a recent phrase-based statistical MT system called PHARAOH for tactical generation. We also showed that the WASP semantic parsing algorithm can be inverted to produce a tactical generation system called WASP^{-1}. This approach allows the same learned grammar to be used for both parsing and generation. Also it allows the chart parser in WASP to be used for generation with minimal modifications. While reasonably effective, both PHARAOH and WASP^{-1} can be substantially improved by borrowing ideas from each other. The resulting hybrid systems, PHARAOH++ and WASP^{-1}++, were shown to be much more robust and accurate, based on automatic and human evaluations in the GEOQUERY and ROBOCUP domains. In particular, the SCFG-based hybrid system WASP^{-1}, produced by inverting WASP and incorporating PHARAOH's probabilistic model, was shown to be the best overall among the four proposed systems.

Lastly, we extended the WASP^{-1}++ tactical generation algorithm to handle predicate logic (Chapter 6). The resulting algorithm, λ-WASP^{-1}++, shares the same underlying λ-SCFG grammar with λ-WASP, and the same probabilistic model with WASP^{-1}++. We presented a chart generation algorithm that can handle input logical forms. Experiments showed that λ-WASP^{-1}++ is competitive compared to other MT-based generators, especially when training data is scarce.

Overall, the research presented in this thesis has made significant contributions to natural language processing in the following two aspects. First, while the use of a single grammar for both parsing and generation has long been advocated for its elegance, and several implementations of this idea have already existed (Section 2.3.1), our work is the first attempt to use the same *automatically-learned* grammar for both parsing and generation. Our WASP-based parsers and generators acquire all of their linguistic knowledge from annotated corpora, unlike other existing systems that require manually-constructed grammars and lexicons (e.g. Carroll and Oepen, 2005). Therefore, our WASP-based systems require much less tedious domain-specific knowledge engineering, and can be easily ported to other languages and application domains.

Second, while our MT-based parsers and generators have only been empirically tested in restricted domains such as GEOQUERY, our work represents an important step toward broad-domain natural language understanding and generation. There is no reason to believe that similar MT-based approaches cannot be used for understanding and generating unrestricted texts, as statistical MT systems with massive amounts of training data have already demonstrated the ability to translate between a wide variety of languages. As argued in Chapter 7, there are three major challenges to solve: (1) devising a suitable MRL for a broad array of applications, such as question answering and interlingual MT, (2) acquiring a knowledge repository that captures all important concepts in the world, and (3) gathering enough training data for effective statistical learning. Solving these problems will require major breakthroughs in areas such as knowledge representation and reasoning, machine learning, natural language processing, and data mining. However, we expect that statistical MT methods will still be relevant because the basic problem of mapping NL expressions to concepts will remain the same. We can see plenty there that needs to be done, but at least we can see the road ahead.

Appendix

Appendix A

Grammars for Meaning Representation Languages

This appendix describes the grammars for all of the formal MRLs considered in this thesis, namely the GEOQUERY logical query language, the GEOQUERY functional query language (FUNQL), and CLANG (Section 2.1). These formal MRL grammars are used to train various semantic parsers and tactical generators, including all WASP-based systems and the PHARAOH++ tactical generator (Section 5.3.1).

A.1 The GEOQUERY Logical Query Language

The GEOQUERY logical query language was devised by Zelle (1995, Sec. 7.3) for querying a U.S. geography database called GEOQUERY. Since the database was written in Prolog, the query language is basically first-order Prolog logical forms, augmented with several meta-predicates for dealing with quantification.

There are 14 different non-terminal symbols in this grammar, of which QUERY is the start symbol. The following non-terminal symbols are for entities referenced in the GEOQUERY database:

Entity types	*Non-terminals*	*Sample productions*
City names	CITYNAME	CITYNAME → `austin`
Country names	COUNTRYNAME	COUNTRYNAME → `usa`
Place names (lakes, mountains, etc.)	PLACENAME	PLACENAME → `tahoe`
River names	RIVERNAME	RIVERNAME → `mississippi`
State abbreviations	STATEABBREV	STATEABBREV → `tx`
State names	STATENAME	STATENAME → `texas`
Numbers	NUM	NUM → `0`

The following non-terminals are used to disambiguate between entities that share the same name (e.g. the state of Mississippi and the Mississippi river). Note the corresponding Prolog functors (e.g. `stateid` and `riverid`):

Entity types	*Non-terminals*	*Productions*
Cities	CITY	CITY → `cityid(`CITYNAME`,`STATEABBREV`)`
		CITY → `cityid(`CITYNAME`,_)`
Countries	COUNTRY	COUNTRY → `countryid(`COUNTRYNAME`)`
Places	PLACE	PLACE → `placeid(`PLACENAME`)`
Rivers	RIVER	RIVER → `riverid(`RIVERNAME`)`
States	STATE	STATE → `stateid(`STATENAME`)`

The FORM non-terminal (short for "formula") is for the following first-order predicates, which provide most of the expressiveness of the GEOQUERY language. Note that $x_1, x_2, \ldots$ are logical variables that denote entities:

Productions	*Meaning of predicates*
FORM → `capital(`x_1`)`	x_1 is a capital (city).
FORM → `city(`x_1`)`	x_1 is a city.
FORM → `country(`x_1`)`	x_1 is a country.
FORM → `lake(`x_1`)`	x_1 is a lake.
FORM → `major(`x_1`)`	x_1 is major (as in a *major* city or a *major* river).
FORM → `mountain(`x_1`)`	x_1 is a mountain.

Productions	*Meaning of predicates*
FORM $\rightarrow$ `place`(x_1)	x_1 is a place.
FORM $\rightarrow$ `river`(x_1)	x_1 is a river.
FORM $\rightarrow$ `state`(x_1)	x_1 is a state.
FORM $\rightarrow$ `area`(x_1,x_2)	The area of x_1 is x_2.
FORM $\rightarrow$ `capital`(x_1,x_2)	The capital of x_1 is x_2.
FORM $\rightarrow$ `density`(x_1,x_2)	The population density of x_1 is x_2.
FORM $\rightarrow$ `elevation`(x_1,x_2)	The elevation of x_1 is x_2.
FORM $\rightarrow$ `elevation`(x_1,NUM)	The elevation of x_1 is NUM.
FORM $\rightarrow$ `high_point`(x_1,x_2)	The highest point of x_1 is x_2.
FORM $\rightarrow$ `higher`(x_1,x_2)	The elevation of x_1 is greater than that of x_2.
FORM $\rightarrow$ `len`(x_1,x_2)	The length of x_1 is x_2.
FORM $\rightarrow$ `loc`(x_1,x_2)	x_1 is located in x_2.
FORM $\rightarrow$ `longer`(x_1,x_2)	The length of x_1 is greater than that of x_2.
FORM $\rightarrow$ `low_point`(x_1,x_2)	The lowest point of x_1 is x_2.
FORM $\rightarrow$ `lower`(x_1,x_2)	The elevation of x_1 is less than that of x_2.
FORM $\rightarrow$ `next_to`(x_1,x_2)	x_1 is adjacent to x_2.
FORM $\rightarrow$ `population`(x_1,x_2)	The population of x_1 is x_2.
FORM $\rightarrow$ `size`(x_1,x_2)	The size of x_1 is x_2.
FORM $\rightarrow$ `traverse`(x_1,x_2)	x_1 traverses x_2.

The following m-tuples are used to constrain the combinations of entity types that the arguments of a m-place predicate can denote. See Section 4.2.5 for how to use these m-tuples for type checking:

Predicates	*Possible entity types for logical variables*
`capital`(x_1)	(CITY), (PLACE)
`city`(x_1)	(CITY)
`country`(x_1)	(COUNTRY)
`lake`(x_1)	(PLACE), (LAKE)
`major`(x_1)	(CITY), (LAKE), (RIVER)

Predicates	*Possible entity types for logical variables*
`mountain(`x_1`)`	(PLACE), (MOUNTAIN)
`place(`x_1`)`	(PLACE), (LAKE), (MOUNTAIN)
`river(`x_1`)`	(RIVER)
`state(`x_1`)`	(STATE)
`area(`x_1`,`x_2`)`	(CITY, NUM), (COUNTRY, NUM), (STATE, NUM)
`capital(`x_1`,`x_2`)`	(STATE, CITY)
`density(`x_1`,`x_2`)`	(CITY, NUM), (COUNTRY, NUM), (STATE, NUM)
`elevation(`x_1`,`x_2`)`	(PLACE, NUM), (MOUNTAIN, NUM)
`elevation(`x_1`,`NUM`)`	(PLACE), (MOUNTAIN)
`high_point(`x_1`,`x_2`)`	(COUNTRY, PLACE), (COUNTRY, MOUNTAIN), (STATE, PLACE), (STATE, MOUNTAIN)
`higher(`x_1`,`x_2`)`	(PLACE, PLACE), (PLACE, MOUNTAIN), (MOUNTAIN, PLACE), (MOUNTAIN, MOUNTAIN)
`len(`x_1`,`x_2`)`	(RIVER, NUM)
`loc(`x_1`,`x_2`)`	(CITY, COUNTRY), (PLACE, COUNTRY), (LAKE, COUNTRY), (MOUNTAIN, COUNTRY), (RIVER, COUNTRY), (STATE, COUNTRY), (CITY, STATE), (PLACE, STATE), (LAKE, STATE), (MOUNTAIN, STATE), (RIVER, STATE), (PLACE, CITY)
`longer(`x_1`,`x_2`)`	(RIVER, RIVER)
`low_point(`x_1`,`x_2`)`	(COUNTRY, PLACE), (COUNTRY, MOUNTAIN), (STATE, PLACE), (STATE, MOUNTAIN)
`lower(`x_1`,`x_2`)`	(PLACE, PLACE), (PLACE, MOUNTAIN), (MOUNTAIN, PLACE), (MOUNTAIN, MOUNTAIN)
`next_to(`x_1`,`x_2`)`	(STATE, RIVER), (STATE, STATE)
`population(`x_1`,`x_2`)`	(CITY, NUM), (COUNTRY, NUM), (STATE, NUM)
`size(`x_1`,`x_2`)`	(CITY, NUM), (COUNTRY, NUM), (PLACE, NUM), (LAKE, NUM), (MOUNTAIN, NUM), (RIVER, NUM), (STATE, NUM)

Predicates	*Possible entity types for logical variables*
`traverse(`x_1`,`x_2`)`	(RIVER, CITY), (RIVER, COUNTRY), (RIVER, STATE)

In addition, the `equal` predicate is used to equate logical variables to ground terms, e.g. `equal(`x_1`,cityid(austin,tx))`:

Productions	*Possible entity types for logical variables*
FORM $\rightarrow$ `equal(`x_1`,`CITY`)`	(CITY)
FORM $\rightarrow$ `equal(`x_1`,`COUNTRY`)`	(COUNTRY)
FORM $\rightarrow$ `equal(`x_1`,`PLACE`)`	(PLACE), (LAKE), (MOUNTAIN)
FORM $\rightarrow$ `equal(`x_1`,`RIVER`)`	(RIVER)
FORM $\rightarrow$ `equal(`x_1`,`STATE`)`	(STATE)

Another important production is the conjunction operator (`,`), which is used to form conjunctions of formulas:

FORM $\rightarrow$ `(`FORM`,`FORM`)`

The `not` operator is used to form negations:

FORM $\rightarrow$ `not(`FORM`)`

The FORM non-terminal is also for the following meta-predicates, which take conjunctive goals as their arguments:

Productions	*Meaning of meta-predicates*
FORM $\rightarrow$ `largest(`x_1`,`FORM`)`	The goal denoted by FORM produces only the solution maximizing the size of x_1.
FORM $\rightarrow$ `smallest(`x_1`,`FORM`)`	The goal denoted by FORM produces only the solution minimizing the size of x_1.
FORM $\rightarrow$ `highest(`x_1`,`FORM`)`	Analogous to `largest` (with elevation).
FORM $\rightarrow$ `lowest(`x_1`,`FORM`)`	Analogous to `smallest` (with elevation).

Productions	*Meaning of meta-predicates*
FORM → `longest`(x_1,FORM)	Analogous to `largest` (with length).
FORM → `shortest`(x_1,FORM)	Analogous to `smallest` (with length).
FORM → `count`(x_1,FORM,x_2)	x_2 is the number of bindings for x_1 satisfying the goal denoted by FORM.
FORM → `sum`(x_1,FORM,x_2)	x_2 is the sum of all bindings for x_1 satisfying the goal denoted by FORM.
FORM → `most`(x_1,x_2,FORM)	The goal denoted by FORM produces only the x_1 maximizing the count of x_2.
FORM → `fewest`(x_1,x_2,FORM)	The goal denoted by FORM produces only the x_1 minimizing the count of x_2.

Below are the corresponding m-tuples of entity types for type checking:

Meta-predicates	*Possible entity types for logical variables*
`largest`(x_1,FORM)	(CITY), (PLACE), (LAKE), (MOUNTAIN), (NUM), (RIVER), (STATE)
`smallest`(x_1,FORM)	(CITY), (PLACE), (LAKE), (MOUNTAIN), (NUM), (RIVER), (STATE)
`highest`(x_1,FORM)	(PLACE), (MOUNTAIN)
`lowest`(x_1,FORM)	(PLACE), (MOUNTAIN)
`longest`(x_1,FORM)	(RIVER)
`shortest`(x_1,FORM)	(RIVER)
`count`(x_1,FORM,x_2)	($*$, NUM)
`sum`(x_1,FORM,x_2)	(NUM, NUM)
`most`(x_1,x_2,FORM)	($*$, $*$)
`fewest`(x_1,x_2,FORM)	($*$, $*$)

In the above table, $*$ denotes any of these entity types: CITY, COUNTRY, PLACE, LAKE, MOUNTAIN, NUM, RIVER, STATE.

Finally, the start symbol, QUERY, is reserved for the `answer` meta-predicate, which serves as a wrapper for query goals (denoted by FORM):

QUERY $\rightarrow$ `answer(`x_1`,`FORM`)`

Here x_1 is the logical variable whose binding is of interest (i.e. answers the question posed). x_1 can denote entities of any type ($*$).

A.2 The GEOQUERY Functional Query Language

For semantic parsers and tactical generators that cannot handle logical variables (e.g. WASP, PHARAOH++, WASP^{-1}++), a variable-free, functional query language called FUNQL has been devised for the GEOQUERY domain (Kate et al., 2005). Below is a sample FUNQL query, together with its corresponding Prolog logical form:

What are the cities in Texas?
FUNQL: `answer(city(loc_2(stateid(texas))))`
Prolog logical form: `answer(`x_1`,(city(`x_1`),loc(`x_1`,`x_2`),`
`equal(`x_2`,stateid(texas))))`

In Section 2.1, we noted that FUNQL predicates can have a set-theoretic interpretation. For example, the term `stateid(texas)` denotes a singleton set that consists of the Texas state, and `loc_2(stateid(texas))` denotes the set of entities located in the Texas state, and so on. Here we present another interpretation of FUNQL based on the lambda calculus. Under this interpretation, each FUNQL predicate is a shorthand for a λ-function, which can be used to translate FUNQL expressions into the GEOQUERY logical query language through function application. For example, the FUNQL predicate `stateid` denotes the λ-function $\lambda n.\lambda x_1.$`equal(`x_1`,stateid(`n`))`. Hence by function application, the FUNQL term `stateid(texas)` is equivalent to the following logical form in the GEOQUERY logical query language:

λx_1.`equal(`x_1`,stateid(texas))`

Also since the FUNQL predicate `loc_2` denotes $\lambda p.\lambda x_1$.`(loc(`x_1`,`x_2`),`$p(x_2)$`)`, the FUNQL term `loc_2(stateid(texas))` is equivalent to:

λx_1.`loc(`x_1`,`x_2`),equal(`x_2`,stateid(texas)))`

There are 13 different non-terminal symbols in the FUNQL grammar. All of them are from the GEOQUERY logical query language. Only the FORM non-terminal is not used in FUNQL. QUERY is the start symbol in the FUNQL grammar.

Below are the FUNQL productions for named entities and numbers, which are identical to those in the GEOQUERY logical query language:

Entity types	*Sample productions*	*Corresponding λ-functions*
City names	CITYNAME → `austin`	`austin`
Country names	COUNTRYNAME → `usa`	`usa`
Place names	PLACENAME → `tahoe`	`tahoe`
River names	RIVERNAME → `mississippi`	`mississippi`
State abbreviations	STATEABBREV → `tx`	`tx`
State names	STATENAME → `texas`	`texas`
Numbers	NUM → `0`	`0`

The rest of the FUNQL productions are as follows:

Productions	*Corresponding λ-functions*
CITY → `cityid(`CITYNAME`,`STATEABBREV`)`	$\lambda n.\lambda a.\lambda x_1$.`equal(`$x_1$`,cityid(`$n$`,`$a$`))`
CITY → `cityid(`CITYNAME`,_)`	$\lambda n.\lambda x_1$.`equal(`x_1`,cityid(`n`,_))`
COUNTRY → `countryid(`COUNTRYNAME`)`	$\lambda n.\lambda x_1$.`equal(`x_1`,countryid(`n`))`
PLACE → `placeid(`PLACENAME`)`	$\lambda n.\lambda x_1$.`equal(`x_1`,placeid(`n`))`
RIVER → `riverid(`RIVERNAME`)`	$\lambda n.\lambda x_1$.`equal(`x_1`,riverid(`n`))`
STATE → `stateid(`STATENAME`)`	$\lambda n.\lambda x_1$.`equal(`x_1`,stateid(`n`))`

Productions	*Corresponding λ-functions*
CITY → `capital(all)`	λx_1.`capital(`x_1`)`
CITY → `city(all)`	λx_1.`city(`x_1`)`
COUNTRY → `country(all)`	λx_1.`country(`x_1`)`
PLACE → `lake(all)`	λx_1.`lake(`x_1`)`
PLACE → `mountain(all)`	λx_1.`mountain(`x_1`)`
PLACE → `place(all)`	λx_1.`place(`x_1`)`
RIVER → `river(all)`	λx_1.`river(`x_1`)`
STATE → `state(all)`	λx_1.`state(`x_1`)`
CITY → `capital(`CITY`)`	$\lambda p.\lambda x_1$.`(capital(`x_1`),`$p(x_1)$`)`
CITY → `capital(`PLACE`)`	$\lambda p.\lambda x_1$.`(capital(`x_1`),`$p(x_1)$`)`
CITY → `city(`CITY`)`	$\lambda p.\lambda x_1$.`(city(`x_1`),`$p(x_1)$`)`
PLACE → `lake(`PLACE`)`	$\lambda p.\lambda x_1$.`(lake(`x_1`),`$p(x_1)$`)`
CITY → `major(`CITY`)`	$\lambda p.\lambda x_1$.`(major(`x_1`),`$p(x_1)$`)`
PLACE → `major(`PLACE`)`	$\lambda p.\lambda x_1$.`(major(`x_1`),`$p(x_1)$`)`
RIVER → `major(`RIVER`)`	$\lambda p.\lambda x_1$.`(major(`x_1`),`$p(x_1)$`)`
PLACE → `mountain(`PLACE`)`	$\lambda p.\lambda x_1$.`(mountain(`x_1`),`$p(x_1)$`)`
PLACE → `place(`PLACE`)`	$\lambda p.\lambda x_1$.`(place(`x_1`),`$p(x_1)$`)`
RIVER → `river(`RIVER`)`	$\lambda p.\lambda x_1$.`(river(`x_1`),`$p(x_1)$`)`
STATE → `state(`STATE`)`	$\lambda p.\lambda x_1$.`(state(`x_1`),`$p(x_1)$`)`
NUM → `area_1(`CITY`)`	$\lambda p.\lambda x_1$.`(area(`x_2`,`x_1`),`$p(x_2)$`)`
NUM → `area_1(`COUNTRY`)`	$\lambda p.\lambda x_1$.`(area(`x_2`,`x_1`),`$p(x_2)$`)`
NUM → `area_1(`PLACE`)`	$\lambda p.\lambda x_1$.`(area(`x_2`,`x_1`),`$p(x_2)$`)`
NUM → `area_1(`STATE`)`	$\lambda p.\lambda x_1$.`(area(`x_2`,`x_1`),`$p(x_2)$`)`
CITY → `capital_1(`COUNTRY`)`	$\lambda p.\lambda x_1$.`(capital(`x_2`,`x_1`),`$p(x_2)$`)`
CITY → `capital_1(`STATE`)`	$\lambda p.\lambda x_1$.`(capital(`x_2`,`x_1`),`$p(x_2)$`)`
STATE → `capital_2(`CITY`)`	$\lambda p.\lambda x_1$.`(capital(`x_1`,`x_2`),`$p(x_2)$`)`
NUM → `density_1(`CITY`)`	$\lambda p.\lambda x_1$.`(density(`x_2`,`x_1`),`$p(x_2)$`)`
NUM → `density_1(`COUNTRY`)`	$\lambda p.\lambda x_1$.`(density(`x_2`,`x_1`),`$p(x_2)$`)`
NUM → `density_1(`STATE`)`	$\lambda p.\lambda x_1$.`(density(`x_2`,`x_1`),`$p(x_2)$`)`
NUM → `elevation_1(`PLACE`)`	$\lambda p.\lambda x_1$.`(elevation(`x_2`,`x_1`),`$p(x_2)$`)`
PLACE → `elevation_2(`NUM`)`	$\lambda n.\lambda x_1$.`elevation(`x_1`,`n`)`

Productions	*Corresponding λ-functions*
PLACE → high_point_1 (STATE)	$\lambda p.\lambda x_1.$(high_point (x_2, x_1), $p(x_2)$)
STATE → high_point_2 (PLACE)	$\lambda p.\lambda x_1.$(high_point (x_1, x_2), $p(x_2)$)
PLACE → higher_2 (PLACE)	$\lambda p.\lambda x_1.$(higher (x_1, x_2), $p(x_2)$)
NUM → len (RIVER)	$\lambda p.\lambda x_1.$(len (x_2, x_1), $p(x_2)$)
CITY → loc_1 (PLACE)	$\lambda p.\lambda x_1.$(loc (x_2, x_1), $p(x_2)$)
COUNTRY → loc_1 (CITY)	$\lambda p.\lambda x_1.$(loc (x_2, x_1), $p(x_2)$)
COUNTRY → loc_1 (PLACE)	$\lambda p.\lambda x_1.$(loc (x_2, x_1), $p(x_2)$)
COUNTRY → loc_1 (RIVER)	$\lambda p.\lambda x_1.$(loc (x_2, x_1), $p(x_2)$)
COUNTRY → loc_1 (STATE)	$\lambda p.\lambda x_1.$(loc (x_2, x_1), $p(x_2)$)
STATE → loc_1 (CITY)	$\lambda p.\lambda x_1.$(loc (x_2, x_1), $p(x_2)$)
STATE → loc_1 (PLACE)	$\lambda p.\lambda x_1.$(loc (x_2, x_1), $p(x_2)$)
STATE → loc_1 (RIVER)	$\lambda p.\lambda x_1.$(loc (x_2, x_1), $p(x_2)$)
CITY → loc_2 (COUNTRY)	$\lambda p.\lambda x_1.$(loc (x_1, x_2), $p(x_2)$)
CITY → loc_2 (STATE)	$\lambda p.\lambda x_1.$(loc (x_1, x_2), $p(x_2)$)
PLACE → loc_2 (CITY)	$\lambda p.\lambda x_1.$(loc (x_1, x_2), $p(x_2)$)
PLACE → loc_2 (STATE)	$\lambda p.\lambda x_1.$(loc (x_1, x_2), $p(x_2)$)
PLACE → loc_2 (COUNTRY)	$\lambda p.\lambda x_1.$(loc (x_1, x_2), $p(x_2)$)
RIVER → loc_2 (COUNTRY)	$\lambda p.\lambda x_1.$(loc (x_1, x_2), $p(x_2)$)
RIVER → loc_2 (STATE)	$\lambda p.\lambda x_1.$(loc (x_1, x_2), $p(x_2)$)
STATE → loc_2 (COUNTRY)	$\lambda p.\lambda x_1.$(loc (x_1, x_2), $p(x_2)$)
RIVER → longer (RIVER)	$\lambda p.\lambda x_1.$(longer (x_1, x_2), $p(x_2)$)
PLACE → lower_2 (PLACE)	$\lambda p.\lambda x_1.$(lower (x_1, x_2), $p(x_2)$)
STATE → next_to_1 (STATE)	$\lambda p.\lambda x_1.$(next_to (x_2, x_1), $p(x_2)$)
STATE → next_to_2 (STATE)	$\lambda p.\lambda x_1.$(next_to (x_1, x_2), $p(x_2)$)
STATE → next_to_2 (RIVER)	$\lambda p.\lambda x_1.$(next_to (x_1, x_2), $p(x_2)$)
NUM → population_1 (CITY)	$\lambda p.\lambda x_1.$(population (x_2, x_1), $p(x_2)$)
NUM → population_1 (COUNTRY)	$\lambda p.\lambda x_1.$(population (x_2, x_1), $p(x_2)$)
NUM → population_1 (STATE)	$\lambda p.\lambda x_1.$(population (x_2, x_1), $p(x_2)$)
NUM → size (CITY)	$\lambda p.\lambda x_1.$(size (x_2, x_1), $p(x_2)$)
NUM → size (COUNTRY)	$\lambda p.\lambda x_1.$(size (x_2, x_1), $p(x_2)$)
NUM → size (STATE)	$\lambda p.\lambda x_1.$(size (x_2, x_1), $p(x_2)$)

Productions	*Corresponding λ-functions*
CITY → traverse_1(RIVER)	$\lambda p.\lambda x_1.$(traverse(x_2,x_1),$p(x_2)$)
COUNTRY → traverse_1(RIVER)	$\lambda p.\lambda x_1.$(traverse(x_2,x_1),$p(x_2)$)
STATE → traverse_1(RIVER)	$\lambda p.\lambda x_1.$(traverse(x_2,x_1),$p(x_2)$)
RIVER → traverse_2(CITY)	$\lambda p.\lambda x_1.$(traverse(x_1,x_2),$p(x_2)$)
RIVER → traverse_2(COUNTRY)	$\lambda p.\lambda x_1.$(traverse(x_1,x_2),$p(x_2)$)
RIVER → traverse_2(STATE)	$\lambda p.\lambda x_1.$(traverse(x_1,x_2),$p(x_2)$)
CITY → largest(CITY)	$\lambda p.\lambda x_1.$largest(x_1,$p(x_1)$)
PLACE → largest(PLACE)	$\lambda p.\lambda x_1.$largest(x_1,$p(x_1)$)
STATE → largest(STATE)	$\lambda p.\lambda x_1.$largest(x_1,$p(x_1)$)
STATE → largest_one(area_1(STATE))	$\lambda p.\lambda x_1.$largest(x_2, (area(x_1,x_2),$p(x_1)$))
CITY → largest_one(density_1(CITY))	$\lambda p.\lambda x_1.$largest(x_2, (density(x_1,x_2),$p(x_1)$))
STATE → largest_one(density_1(STATE))	$\lambda p.\lambda x_1.$largest(x_2, (density(x_1,x_2),$p(x_1)$))
CITY → largest_one(population_1(CITY))	$\lambda p.\lambda x_1.$largest(x_2, (population(x_1,x_2),$p(x_1)$))
STATE → largest_one(population_1(STATE))	$\lambda p.\lambda x_1.$largest(x_2, (population(x_1,x_2),$p(x_1)$))
CITY → smallest(CITY)	$\lambda p.\lambda x_1.$smallest(x_1,$p(x_1)$)
NUM → smallest(NUM)	$\lambda p.\lambda x_1.$smallest(x_1,$p(x_1)$)
PLACE → smallest(PLACE)	$\lambda p.\lambda x_1.$smallest(x_1,$p(x_1)$)
STATE → smallest(STATE)	$\lambda p.\lambda x_1.$smallest(x_1,$p(x_1)$)
STATE → smallest_one(area_1(STATE))	$\lambda p.\lambda x_1.$smallest(x_2, (area(x_1,x_2),$p(x_1)$))
STATE → smallest_one(density_1(STATE))	$\lambda p.\lambda x_1.$smallest(x_2, (density(x_1,x_2),$p(x_1)$))
CITY → smallest_one(population_1(CITY))	$\lambda p.\lambda x_1.$smallest(x_2, (population(x_1,x_2),$p(x_1)$))
STATE → smallest_one(population_1(STATE))	$\lambda p.\lambda x_1.$smallest(x_2, (population(x_1,x_2),$p(x_1)$))

Productions	*Corresponding λ-functions*
PLACE $\rightarrow$ `highest(`PLACE`)`	$\lambda p.\lambda x_1.$`highest(`x_1`,`$p(x_1)$`)`
PLACE $\rightarrow$ `lowest(`PLACE`)`	$\lambda p.\lambda x_1.$`lowest(`x_1`,`$p(x_1)$`)`
RIVER $\rightarrow$ `longest(`RIVER`)`	$\lambda p.\lambda x_1.$`longest(`x_1`,`$p(x_1)$`)`
RIVER $\rightarrow$ `shortest(`RIVER`)`	$\lambda p.\lambda x_1.$`shortest(`x_1`,`$p(x_1)$`)`
NUM $\rightarrow$ `count(`CITY`)`	$\lambda p.\lambda x_1.$`count(`x_2`,`$p(x_2)$`,`x_1`)`
NUM $\rightarrow$ `count(`PLACE`)`	$\lambda p.\lambda x_1.$`count(`x_2`,`$p(x_2)$`,`x_1`)`
NUM $\rightarrow$ `count(`RIVER`)`	$\lambda p.\lambda x_1.$`count(`x_2`,`$p(x_2)$`,`x_1`)`
NUM $\rightarrow$ `count(`STATE`)`	$\lambda p.\lambda x_1.$`count(`x_2`,`$p(x_2)$`,`x_1`)`
NUM $\rightarrow$ `sum(`NUM`)`	$\lambda p.\lambda x_1.$`sum(`x_2`,`$p(x_2)$`,`x_1`)`
CITY $\rightarrow$ `most(`CITY`)`	$\lambda p'.\lambda x_1.$`most(`x_1`,`x'`,`$p'(x_1)$`)`, where p' contains one and only one free variable, x'
PLACE $\rightarrow$ `most(`PLACE`)`	$\lambda p'.\lambda x_1.$`most(`x_1`,`x'`,`$p'(x_1)$`)`
RIVER $\rightarrow$ `most(`RIVER`)`	$\lambda p'.\lambda x_1.$`most(`x_1`,`x'`,`$p'(x_1)$`)`
STATE $\rightarrow$ `most(`STATE`)`	$\lambda p'.\lambda x_1.$`most(`x_1`,`x'`,`$p'(x_1)$`)`
CITY $\rightarrow$ `fewest(`CITY`)`	$\lambda p'.\lambda x_1.$`fewest(`x_1`,`x'`,`$p'(x_1)$`)`
PLACE $\rightarrow$ `fewest(`PLACE`)`	$\lambda p'.\lambda x_1.$`fewest(`x_1`,`x'`,`$p'(x_1)$`)`
RIVER $\rightarrow$ `fewest(`RIVER`)`	$\lambda p'.\lambda x_1.$`fewest(`x_1`,`x'`,`$p'(x_1)$`)`
STATE $\rightarrow$ `fewest(`STATE`)`	$\lambda p'.\lambda x_1.$`fewest(`x_1`,`x'`,`$p'(x_1)$`)`
CITY $\rightarrow$ `intersection(`CITY`,`CITY`)`	$\lambda p_1.\lambda p_2.\lambda x_1.$`(`$p_1(x_1)$`,`$p_2(x_1)$`)`
PLACE $\rightarrow$ `intersection(`PLACE`,`PLACE`)`	$\lambda p_1.\lambda p_2.\lambda x_1.$`(`$p_1(x_1)$`,`$p_2(x_1)$`)`
RIVER $\rightarrow$ `intersection(`RIVER`,`RIVER`)`	$\lambda p_1.\lambda p_2.\lambda x_1.$`(`$p_1(x_1)$`,`$p_2(x_1)$`)`
STATE $\rightarrow$ `intersection(`STATE`,`STATE`)`	$\lambda p_1.\lambda p_2.\lambda x_1.$`(`$p_1(x_1)$`,`$p_2(x_1)$`)`
CITY $\rightarrow$ `exclude(`CITY`,`CITY`)`	$\lambda p_1.\lambda p_2.\lambda x_1.$`(`$p_1(x_1)$`,not(`$p_2(x_1)$`))`
PLACE $\rightarrow$ `exclude(`PLACE`,`PLACE`)`	$\lambda p_1.\lambda p_2.\lambda x_1.$`(`$p_1(x_1)$`,not(`$p_2(x_1)$`))`
RIVER $\rightarrow$ `exclude(`RIVER`,`RIVER`)`	$\lambda p_1.\lambda p_2.\lambda x_1.$`(`$p_1(x_1)$`,not(`$p_2(x_1)$`))`
STATE $\rightarrow$ `exclude(`STATE`,`STATE`)`	$\lambda p_1.\lambda p_2.\lambda x_1.$`(`$p_1(x_1)$`,not(`$p_2(x_1)$`))`

Productions	*Corresponding λ-functions*
QUERY → `answer(`CITY`)`	λp.`answer(`x_1`,`$p(x_1)$`)`
QUERY → `answer(`COUNTRY`)`	λp.`answer(`x_1`,`$p(x_1)$`)`
QUERY → `answer(`NUM`)`	λp.`answer(`x_1`,`$p(x_1)$`)`
QUERY → `answer(`PLACE`)`	λp.`answer(`x_1`,`$p(x_1)$`)`
QUERY → `answer(`RIVER`)`	λp.`answer(`x_1`,`$p(x_1)$`)`
QUERY → `answer(`STATE`)`	λp.`answer(`x_1`,`$p(x_1)$`)`

A.3 CLANG: The ROBOCUP Coach Language

In the ROBOCUP Coach Competition, teams compete to provide effective instructions to advice-taking agents in the simulated soccer domain. Coaching instructions are provided in a formal coach language called CLANG (Chen et al., 2003, Sec. 7.7).

The CLANG grammar described here basically follows the one described in Chen et al. (2003). We have slightly modified CLANG to introduce a few concepts that are not easily describable in the original CLANG language. These new constructs are marked with asterisks (*).

In CLANG, coaching instructions come in the form of *if-then rules*. Each if-then rule consists of a *condition* and a *directive*:

RULE → `(`CONDITION DIRECTIVE`)`

Possible conditions are:

Productions	*Meaning of predicates*
CONDITION → `(true)`	Always true.
CONDITION → `(false)`	Always false.

Productions	*Meaning of predicates*
CONDITION → `(ppos` PLAYER UNUM$_1$ UNUM$_2$ REGION`)`	At least UNUM$_1$ and at most UNUM$_2$ of PLAYER is in REGION.
CONDITION → `(ppos-any` PLAYER REGION`)`*	Some of PLAYER is in REGION.
CONDITION → `(ppos-none our` REGION`)`*	None of our players is in REGION.
CONDITION → `(ppos-none opp` REGION`)`*	None of the opponents is in REGION.
CONDITION → `(bpos` REGION`)`	The ball is in REGION.
CONDITION → `(bowner` PLAYER`)`	PLAYER owns the ball.
CONDITION → `(playm bko)`	Specific play modes (Chen et al., 2003).
CONDITION → `(playm time_over)`	
CONDITION → `(playm play_on)`	
CONDITION → `(playm ko_our)`	
CONDITION → `(playm ko_opp)`	
CONDITION → `(playm ki_our)`	
CONDITION → `(playm ki_opp)`	
CONDITION → `(playm fk_our)`	
CONDITION → `(playm fk_opp)`	
CONDITION → `(playm ck_our)`	
CONDITION → `(playm ck_opp)`	
CONDITION → `(playm gk_our)`	
CONDITION → `(playm gk_opp)`	
CONDITION → `(playm gc_our)`	
CONDITION → `(playm gc_opp)`	
CONDITION → `(playm ag_our)`	
CONDITION → `(playm ag_opp)`	
CONDITION → `"`IDENT`"`	Condition named IDENT. See `definec`.
CONDITION → `(<` NUM$_1$ NUM$_2$`)`	NUM$_1$ is smaller than NUM$_2$. Both NUM$_1$ and NUM$_2$ can be identifiers.
CONDITION → `(>` NUM$_1$ NUM$_2$`)`	NUM$_1$ is greater than NUM$_2$.
CONDITION → `(<=` NUM$_1$ NUM$_2$`)`	NUM$_1$ is not greater than NUM$_2$.
CONDITION → `(==` NUM$_1$ NUM$_2$`)`	NUM$_1$ is equal to NUM$_2$.
CONDITION → `(>=` NUM$_1$ NUM$_2$`)`	NUM$_1$ is not smaller than NUM$_2$.

Productions	*Meaning of predicates*
CONDITION → `(!=` NUM_1 NUM_2`)`	NUM_1 is not equal to NUM_2.
CONDITION → `(and` CONDITION_1 CONDITION_2`)`	CONDITION_1 and CONDITION_2.
CONDITION → `(or` CONDITION_1 CONDITION_2`)`	CONDITION_1 or CONDITION_2.
CONDITION → `(not` CONDITION`)`	CONDITION is not true.

Directives are lists of actions for individual players to take:

Productions	*Meaning of predicates*
DIRECTIVE → `(do` PLAYER ACTION`)`	PLAYER should take ACTION.
DIRECTIVE → `(dont` PLAYER ACTION`)`	PLAYER should avoid taking ACTION.

Possible actions are:

Productions	*Meaning of predicates*
ACTION → `(pos` REGION`)`	Go to REGION.
ACTION → `(home` REGION`)`	Set default position to REGION.
ACTION → `(mark` PLAYER`)`	Mark PLAYER (usually opponents).
ACTION → `(markl` REGION`)`	Mark the passing lane from current ball position to REGION.
ACTION → `(markl` PLAYER`)`	Mark the passing lane from current ball position to position of PLAYER (usually opponents).
ACTION → `(oline` REGION`)`	Set offside-trap line to REGION.
ACTION → `(pass` REGION`)`	Pass the ball to REGION.
ACTION → `(pass` PLAYER`)`	Pass the ball to PLAYER.
ACTION → `(dribble` REGION`)`	Dribble the ball to REGION.
ACTION → `(clear` REGION`)`	Clear the ball to REGION.
ACTION → `(shoot)`	Shoot the ball.
ACTION → `(hold)`	Hold the ball.
ACTION → `(intercept)`	Intercept the ball.
ACTION → `(tackle` PLAYER`)`	Tackle PLAYER.

The following productions are for specifying players: (UNUM stands for "uniform numbers", i.e. 1 to 11)

Productions	*Meaning of predicates*
PLAYER → `(player our` {UNUM}`)`**	Our player UNUM.
PLAYER → `(player our` {UNUM_1 UNUM_2}`)`**	Our players UNUM_1 and UNUM_2.
PLAYER → `(player our` {UNUM_1 UNUM_2 UNUM_3}`)`**	Our players UNUM_1, UNUM_2 and UNUM_3.
PLAYER → `(player our` {UNUM_1 UNUM_2 UNUM_3 UNUM_4}`)`**	Our players UNUM_1, UNUM_2, UNUM_3 and UNUM_4.
PLAYER → `(player opp` {UNUM}`)`**	Opponent player UNUM.
PLAYER → `(player our` {`0`}`)`**	Our team.
PLAYER → `(player opp` {`0`}`)`**	Opponent's team.
PLAYER → `(player-range our` UNUM_1 UNUM_2`)`*	Our players UNUM_1 to UNUM_2.
PLAYER → `(player-range opp` UNUM_1 UNUM_2`)`*	Opponent players UNUM_1 to UNUM_2.
PLAYER → `(player-except our` {UNUM}`)`*	Our team except player UNUM
PLAYER → `(player-except opp` {UNUM}`)`*	Opponent's team except player UNUM

Productions marked with double asterisks (**) are slight variations of existing constructs in the original CLANG grammar (e.g. as in `(bowner our` {`4`}`)`). The new `player` predicate is introduced for uniformity. To specify regions, we can use the following productions:

Productions	*Meaning of predicates*
REGION → POINT	A POINT.
REGION → `(rec` POINT_1 POINT_2`)`	A rectangle with opposite corners POINT_1 and POINT_2.
REGION → `(tri` POINT_1 POINT_2 POINT_3`)`	A triangle with corners POINT_1, POINT_2 and POINT_3.

Productions	*Meaning of predicates*
REGION → `(arc` POINT NUM_1 NUM_2 NUM_3 NUM_4`)`	A donut arc (Chen et al., 2003).
REGION → `(circle` POINT NUM`)`*	A circle of center POINT and radius NUM.
REGION → `(null)`	The empty region.
REGION → `(reg` REGION_1 REGION_2`)`	The union of REGION_1 and REGION_2.
REGION → `(reg-exclude` REGION_1 REGION_2`)`*	REGION_1 excluding REGION_2.
REGION → `(field)`*	The field.
REGION → `(half` TEAM`)`*	The TEAM's half of field. TEAM can be either `our` or `opp`.
REGION → `(penalty-area` TEAM`)`*	The TEAM's penalty area.
REGION → `(goal-area` TEAM`)`*	The TEAM's goal area.
REGION → `(midfield)`*	The midfield.
REGION → `(midfield` TEAM`)`*	The TEAM's midfield.
REGION → `(near-goal-line` TEAM`)`*	Near TEAM's goal line.
REGION → `(from-goal-line` TEAM NUM_1 NUM_2`)`*	NUM_1 to NUM_2 meters from TEAM's goal line.
REGION → `(left` REGION`)`*	The left half of REGION (from our team's perspective).
REGION → `(right` REGION`)`*	The right half of REGION.
REGION → `(left-quarter` REGION`)`*	The left quarter of REGION.
REGION → `(right-quarter` REGION`)`*	The right quarter of REGION.
REGION → `"`IDENT`"`	Region named IDENT. See `definer`.

To specify points, we can use the following productions:

Productions	*Meaning of predicates*
POINT → `(pt` NUM_1 NUM_2`)`	The xy-coordinates $(\text{NUM}_1, \text{NUM}_2)$.
POINT → `(pt ball)`	The current ball position.
POINT → POINT_1 `+` POINT_2	Coordinate-wise addition.
POINT → POINT_1 `-` POINT_2	Coordinate-wise subtraction.
POINT → POINT_1 $\star$ POINT_2	Coordinate-wise multiplication.

Productions	*Meaning of predicates*
POINT → POINT_1 `/` POINT_2	Coordinate-wise division.
POINT → `(pt-with-ball-attraction` POINT_1 POINT_2`)`*	POINT_1 `+ ((pt ball) *` POINT_2`)`.
POINT → `(front-of-goal` TEAM`)`*	Directly in front of TEAM's goal.
POINT → `(from-goal` TEAM NUM`)`*	NUM meters in front of TEAM's goal.

The following CLANG statements can be used to define names for conditions and regions. These names (IDENT) can be used to simplify the definition of if-then rules:

STATEMENT → `(definec "`IDENT`"` CONDITION`)`

STATEMENT → `(definer "`IDENT`"` REGION`)`

Note that an if-then rule is also a CLANG statement:

STATEMENT → RULE

STATEMENT is the start symbol in the CLANG grammar.

Bibliography

Alfred V. Aho and Jeffrey D. Ullman (1969). Properties of syntax directed translations. *Journal of Computer and System Sciences*, 3(3):319–334.

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

James F. Allen (1995). *Natural Language Understanding*. Benjamin/Cummings, Menlo Park, CA, 2nd ed.

Ion Androutsopoulos, Graeme D. Ritchie and Peter Thanisch (1995). Natural language interfaces to databases: An introduction. *Journal of Natural Language Engineering*, 1(1):29–81.

Collin Baker, Michael Ellsworth and Katrin Erk (2007). SemEval-2007 task 19: Frame semantic structure extraction. In *Proceedings of the Fourth International Workshop on Semantic Evaluation (SemEval-2007)*, pp. 99–104. Prague, Czech Republic.

Srinivas Bangalore, Owen Rambow and Steven Whittaker (2000). Evaluation metrics for generation. In *Proceedings of the 1st International Conference on Natural Language Generation (INLG-2000)*, pp. 1–8. Mitzpe Ramon, Israel.

Ken Barker, Bhalchandra Agashe, Shaw-Yi Chaw et al. (2007). Learning by reading: A prototype system, performance baseline and lessons learned. In *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI-2007)*, pp. 280–286. Vancouver, Canada.

John Bateman (1990). Upper modeling: A level of semantics for natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Generation*, pp. 54–61. Dawson, PA.

Samuel Bayer, John Burger, Warren Greiff and Ben Wellner (2004). The MITRE logical form generation system. In *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*. Barcelona, Spain.

Anja Belz and Ehud Reiter (2006). Comparing automatic and human evaluation of NLG systems. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, pp. 313–320. Trento, Italy.

Adam L. Berger, Stephen A. Della Pietra and Vincent J. Della Pietra (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Daniel M. Bikel (2004). Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Patrick Blackburn and Johan Bos (2005). *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI Publications, Stanford, CA.

Borland International (1988). *Turbo Prolog 2.0 Reference Guide*. Borland International, Scotts Valley, CA.

Johan Bos (2005). Towards wide-coverage semantic interpretation. In *Proceedings of the Sixth International Workshop on Computational Semantics (IWCS-2005)*. Tilburg, The Netherlands.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och and Jeffrey Dean (2007). Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007)*, pp. 858–867. Prague, Czech Republic.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer and Paul S. Roossin (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Meredith J. Goldsmith, Jan Hajic, Robert L. Mercer and Surya Mohanty (1993a). But dictionaries are data too. In *Proceedings of the ARPA Workshop on Human Language Technology*, pp. 202–205. Princeton, NJ.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra and Robert L. Mercer (1993b). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312.

Xavier Carreras and Luís Màrquez (2005). Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pp. 152–164. Ann Arbor, MI.

John Carroll, Ann Copestake, Dan Flickinger and Victor Poznański (1999). An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG-1999)*, pp. 86–95. Toulouse, France.

John Carroll and Stephan Oepen (2005). High efficiency realization for a wide-coverage unification grammar. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-2005)*, pp. 165–176. Jeju Island, Korea.

Eugene Charniak (2000). A maximum-entropy-inspired parser. In *Proceedings of the Meeting of the North American Association for Computational Linguistics (NAACL-2000)*, pp. 132–139.

Eugene Charniak and Yorick Wilks, eds. (1976). *Computational Semantics*. North-Holland, Amsterdam.

John Chen, Srinivas Bangalore and K. Vijay-Shanker (2006). Automated extraction of tree-adjoining grammars from treebanks. *Natural Language Engineering*, 12(3):251–299.

Mao Chen, Ehsan Foroughi, Fredrik Heintz et al. (2003). Users manual: RoboCup soccer server manual for soccer server version 7.07 and later. Available at `http://sourceforge.net/projects/sserver/`.

Stanley F. Chen and Ronald Rosenfeld (1999). A Gaussian prior for smoothing maximum entropy model. Technical Report CMU-CS-99-108, School of Computer Science, Carnegie Mellon University.

Colin Cherry and Dekang Lin (2006). Soft syntactic constraints for word alignment through discriminative training. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-2006)*, pp. 105–112. Sydney, Australia.

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

David Chiang, Mona Diab, Nizar Habash, Owen Rambow and Saflullah Shareef (2006). Parsing Arabic dialects. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pp. 369–376. Trento, Italy.

Hai Leong Chieu and Hwee Tou Ng (2003). Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Computational Natural Language Learning (CoNLL-2003)*, pp. 160–163. Edmonton, Canada.

Alonzo Church (1940). A formulation of a simple theory of types. *Journal of Symbolic Logic*, 5:56–68.

Stephen Clark and James R. Curran (2003). Log-linear models for wide-coverage CCG parsing. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP-03)*, pp. 97–105. Sapporo, Japan.

Stephen Clark and James R. Curran (2004). Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-2004)*, pp. 104–111. Barcelona, Spain.

Michael Collins and Terry Koo (2005). Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69.

Michael J. Collins (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97)*, pp. 16–23.

Ann Copestake and Dan Flickinger (2000). An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the Second Conference on Language Resources and Evaluation (LREC-2000)*. Athens, Greece.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein (2001). *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2nd ed.

Simon Corston-Oliver, Michael Gamon, Eric K. Ringger and Robert Moore (2002). An overview of Amalgam: A machine-learned generation module. In *Proceedings of the 2nd International Conference on Natural Language Generation (INLG-2002)*, pp. 33–40. Harriman, NY.

Dick Crouch (2005). Packed rewriting for mapping semantics to KR. In *Proceedings of the Sixth International Workshop on Computational Semantics (IWCS-2005)*. Tilburg, The Netherlands.

John DeNero and Dan Klein (2007). Tailoring word alignments to syntactic machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-2007)*. Prague, Czech Republic.

Yuan Ding, Daniel Gildea and Martha Palmer (2003). An algorithm for word-level alignment of parallel dependency trees. In *Proceedings of the Ninth Machine Translation Summit*, pp. 95–101. New Orleans, LA.

George Doddington (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.

David R. Dowty, Robert E. Wall and Stanley Peters (1981). *Introduction to Montague Semantics*. D. Reidel, Dordrecht, Holland.

Jay Earley (1970). An efficient context-free parsing algorithm. *Communications of the Association for Computing Machinery*, 6(8):451–455.

Michael Elhadad and Jacques Robin (1996). An overview of SURGE: A reusable comprehensive syntactic realization component. Tech. Rep. 96-03, Department of Computer Science, Ben Gurion University, Beer Sheva, Israel.

Katrin Erk (2005). Frame assignment as word sense disambiguation. In *Proceedings of the Sixth International Workshop on Computational Semantics (IWCS-2005)*. Tilburg, The Netherlands.

Katrin Erk and Sebastian Padó (2006). SHALMANESER—a toolchain for shallow semantic parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*. Genoa, Italy.

David Farwell, Stephen Helmreich, Bonnie J. Dorr, Nizar Habash, Florence Reeder, Keith Miller, Lori Levin, Teruko Mitamura, Eduard Hovy, Owen Rambow and Advaith Siddharthan (2004). Interlingual annotation of multilingual text corpora. In *Proceedings of the NAACL-2004 Workshop on Frontiers in Corpus Annotation*, pp. 55–62. Boston, MA.

Christiane D. Fellbaum (1998). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Charles J. Fillmore, Christopher R. Johnson and Miriam R. L. Petruck (2003). Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250.

Heidi J. Fox (2002). Phrasal cohesion and statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*, pp. 304–311. Philadelphia, PA.

Noah S. Friedland, Paul G. Allen, Gavin Matthews et al. (2004). Project Halo: Towards a digital Aristotle. *AI Magazine*, 25(4):29–47.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang and Ignacio Thayer (2006). Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-2006)*, pp. 961–968. Sydney, Australia.

Michel Galley and Kathleen McKeown (2007). Lexicalized Markov grammars for sentence compression. In *Proceedings of North American Chapter of the Association for Computational Linguistics Annual Meeting and Human Language Technology Conference (NAACL-HLT-2007)*, pp. 180–187. Rochester, NY.

Yuqing Gao, Bowen Zhou, Ruhi Sarikaya, Mohamed Afify, Hong-Kwang Kuo, Wei-Zhong Zhu, Yonggang Deng, Charles Prosser, Wei Zhang and Laurent Besacier (2006). IBM MASTOR system: Multilingual automatic speech-to-speech translator. In *Proceedings of the First International Workshop on Medical Speech Translation*, pp. 53–56. New York, NY.

Ruifang Ge and Raymond J. Mooney (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.

Ruifang Ge and Raymond J. Mooney (2006). Discriminative reranking for semantic parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-2006), Poster Sessions*, pp. 263–270. Sydney, Australia.

Daniel Gildea and Daniel Jurafsky (2002). Automated labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

H. Paul Grice (1975). Logic and conversation. In Peter Cole and Jerry Morgan, eds., *Syntax and Semantics 3: Speech Acts*, pp. 41–58. Academic Press, New York.

Dilek Zeynep Hakkani, Gökhan Tür, Kemal Oflazer, Teruko Mitamura and Eric H. Nyberg (1998). An English-to-Turkish interlingual MT system. In *Proceedings of the Third Conference of the Association for Machine Translation in the Americas (AMTA-1998)*, pp. 83–94. Langhorne, PA.

Yulan He and Steve Young (2003). Hidden vector state model for hierarchical semantic parsing. In *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-03)*, pp. 268–271. Hong Kong.

Yulan He and Steve J. Young (2006). Spoken language understanding using the hidden vector state model. *Speech Communication, Special Issue on Spoken Language Understanding for Conversational Systems*, 48(3–4):262–275.

Julia Hockenmaier and Mark Steedman (2002). Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002)*, vol. V, pp. 1974–1981. Las Palmas, Spain.

Liang Huang and David Chiang (2005). Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT-2005)*, pp. 53–64. Vancouver, Canada.

Paul S. Jacobs (1985). PHRED: A generator for natural language interfaces. *Computational Linguistics*, 11(4):219–242.

Rohit J. Kate and Raymond J. Mooney (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-2006)*, pp. 913–920. Sydney, Australia.

Rohit J. Kate, Yuk Wah Wong and Raymond J. Mooney (2005). Learning to transform natural to formal languages. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-2005)*, pp. 1062–1068. Pittsburgh, PA.

Martin Kay (1975). Syntactic processing and functional sentence perspective. In *Theoretical Issues in Natural Language Processing—Supplement to the Proceedings*, pp. 12–15. Cambridge, MA.

Martin Kay (1996). Chart generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL-96)*, pp. 200–204. San Francisco, CA.

Kevin Knight, Ishwar Chander, Matthew Haines, Vasileios Hatzivassiloglou, Eduard Hovy, Masayo Iida, Steve K. Luk, Richard Whitney and Kenji Yamada (1995). Filling knowledge gaps in a broad-coverage machine translation system. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 1390–1397. Montréal, Canada.

Kevin Knight and Jonathan Graehl (2005). An overview of probabilistic tree transducers for natural language processing. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-05)*, pp. 1–25. Mexico City, Mexico.

Kevin Knight and Vasileios Hatzivassiloglou (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.

Kevin Knight and Irene Langkilde (2000). Preserving ambiguities in generation via automata intersection. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pp. 697–702. Austin, TX.

Philipp Koehn and Christof Monz (2006). Manual and automatic evaluation of machine translation between European languages. In *Proceedings on the Workshop on Statistical Machine Translation*, pp. 102–121. New York City, NY.

Philipp Koehn, Franz Josef Och and Daniel Marcu (2003). Statistical phrase-based translation. In *Proceedings of Human Language Technology Conference and North American Association for Computational Linguistics Annual Meeting (HLT-NAACL-2003)*. Edmonton, Canada.

Gregory Kuhlmann, Peter Stone, Raymond J. Mooney and Jude W. Shavlik (2004). Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *Proceedings of the AAAI-04 Workshop on Supervisory Control of Learning and Adaptive Systems*. San Jose, CA.

Roland Kuhn and Renato De Mori (1995). The application of semantic classification trees to natural language understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):449–460.

Irene Langkilde and Kevin Knight (1998). Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and COLING-98 (ACL-COLING-98)*, pp. 704–710. Montréal, Canada.

Irene Langkilde-Geary (2002). An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the 2nd International Conference on Natural Language Generation (INLG-2002)*, pp. 17–24. Harriman, NY.

Maria Lapata and Chris Brew (1999). Using subcategorization to resolve verb class ambiguity. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP-VLC-99)*, pp. 266–274. College Park, MD.

Benoit Lavoie and Owen Rambow (1997). A fast and portable realizer for text generation systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-1997)*, pp. 265–268. Washington, DC.

Yoong Keok Lee and Hwee Tou Ng (2002). An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*, pp. 41–48. Philadelphia, PA.

Roger Levy and Christopher Manning (2004). Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-2004)*, pp. 327–334. Barcelona, Spain.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini and Chris Watkins (2002). Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Klaus Macherey, Franz Josef Och and Hermann Ney (2001). Natural language understanding using statistical machine translation. In *Proceedings of the 7th*

European Conference on Speech Communication and Technology (EuroSpeech-01), pp. 2205–2208. Aalborg, Denmark.

Robert Malouf (2002). A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conference on Computational Natural Language Learning (CoNLL-2002)*, pp. 49–55. Taipei, Taiwan.

Inderjeet Mani (2001). *Automatic Summarization*. John Benjamins, Amsterdam, The Netherlands.

Daniel Marcu and William Wong (2002). A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*, pp. 133–139. Philadelphia, PA.

Jonathan May and Kevin Knight (2007). Syntactic re-alignment models for machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007)*. Prague, Czech Republic.

Scott Miller, Robert Bobrow, Robert Ingria and Richard Schwartz (1994). Hidden understanding models of natural language. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL-94)*, pp. 25–32.

Scott Miller, David Stallard, Robert Bobrow and Richard Schwartz (1996). A fully statistical approach to natural language interfaces. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL-96)*, pp. 55–61. Santa Cruz, CA.

Yusuke Miyao, Takashi Ninomiya and Jun'ichi Tsujii (2004). Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the Penn treebank. In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP-2004)*, pp. 684–693. Sanya City, China.

Yusuke Miyao and Jun'ichi Tsujii (2002). Maximum entropy estimation for feature forests. In *Proceedings of the Human Language Technology Conference (HLT-2002)*. San Diego, CA.

Richard Montague (1970). Universal grammar. *Theoria*, 36:373–398.

Taesun Moon and Jason Baldridge (2007). Part-of-speech tagging for middle English through alignment and projection of parallel diachronic texts. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007)*, pp. 390–399. Prague, Czech Republic.

Robert C. Moore (2002). A complete, efficient sentence-realization algorithm for unification grammar. In *Proceedings of the 2nd International Conference on Natural Language Generation (INLG-2002)*, pp. 41–48. Harriman, NY.

Dragos Stefan Munteanu, Alexander Fraser and Daniel Marcu (2004). Improved machine translation performance via parallel sentence extraction from comparable corpora. In *Proceedings of Human Language Technology Conference and North American Association for Computational Linguistics Annual Meeting (HLT-NAACL-2004)*, pp. 265–272. Boston, MA.

Hiroko Nakanishi, Yusuke Miyao and Jun'ichi Tsujii (2005). Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proceedings of the 9th International Workshop on Parsing Technology (IWPT-2005)*, pp. 93–102. Vancouver, Canada.

NIST (2006). NIST 2006 machine translaiton evaluation official results. Available at `http://www.nist.gov/speech/tests/mt/mt06eval_official_results.html`.

Jorge Nocedal (1980). Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782.

Eric H. Nyberg and Teruko Mitamura (1992). The KANT system: Fast, accurate, high-quality translation in practical domains. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING-1992)*, pp. 1069–1073. Nantes, France.

Franz J. Och, Christoph Tillmann and Hermann Ney (1999). Improved alignment models for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP-VLC-99)*, pp. 20–28. University of Maryland.

Franz Josef Och (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, pp. 160–167. Sapporo, Japan.

Franz Josef Och and Hermann Ney (2000). A comparison of alignment models for statistical machine translation. In *Proceedings of the Eighteenth International Conference on Computational Linguistics (COLING-2000)*, pp. 1086–1090. Saarbrücken, Germany.

Franz Josef Och and Hermann Ney (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och and Hermann Ney (2004). The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–450.

Alice H. Oh and Alex Rudnicky (2000). Stochastic language generation for spoken dialogue systems. In *Proceedings of the ANLP-NAACL-2000 Workshop on Coversational Systems*, pp. 27–32. Seattle, WA.

Martha Palmer, Daniel Gildea and Paul Kingsbury (2005). The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu (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.

Kishore A. Papineni, Salim Roukos and R. Todd Ward (1997). Feature-based language understanding. In *Proceedings of the 5th European Conference on Speech Communication and Technology (EuroSpeech-97)*, pp. 1435–1438. Rhodes, Greece.

Andrew Philpot, Eduard Hovy and Patrick Pantel (2005). The Omega ontology. In *Proceedings of the IJCNLP-2005 Workshop on Ontologies and Lexical Resources (OntoLex-2005)*. Jeju Island, South Korea.

Carl Pollard (1984). *Generalized Phrase Structure Grammars*. Ph.D. thesis, Stanford University.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko and Alexander Yates (2004). Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *Proceedings of the Twentieth International Conference on Computational Linguistics (COLING-2004)*. Geneva, Switzerland.

Ana-Maria Popescu, Oren Etzioni and Henry Kautz (2003). Towards a theory of natural language interfaces to databases. In *Proceedings of the 2003 International Conference on Intelligent User Interfaces (IUI-2003)*, pp. 149–157. ACM, Miami, FL.

Vaughan R. Pratt (1973). A linguistics oriented programming language. In *Proceedings of the Third International Joint Conference on Artificial Intelligence (IJCAI-73)*, pp. 372–382. Stanford, CA.

Patti J. Price (1990). Evaluation of spoken language systems: The ATIS domain. In *Proceedings of the Third DARPA Speech and Natural Language Workshop*, pp. 91–95.

Adwait Ratnaparkhi (1996). A maximum entropy part of speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-96)*, pp. 133–141. Philadelphia, PA.

Stefan Riezler, Tracy King, Ronald Kaplan, Richard Crouch, John Maxwell III and Mark Johnson (2002). Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 271–278. Philadelphia, PA.

Eric Ringger, Michael Gamon, Robert C. Moore, David Rojas, Martine Smets and Simon Corston-Oliver (2004). Linguistically informed statistical models of constituent structure for ordering in sentence realization. In *Proceedings of the Twentieth International Conference on Computational Linguistics (COLING-2004)*, pp. 673–679. Geneva, Switzerland.

Yves Schabes and Aravind K. Joshi (1988). An Earley-type parsing algorithm for tree adjoining grammars. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics (ACL-88)*, pp. 258–269. Buffalo, NY.

William Schuler (2003). Using model-theoretic semantic interpretation to guide statistical parsing and word recognition in a spoken language interface. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, pp. 529–536.

Stephanie Seneff (1992). TINA: A natural language system for spoken language applications. *Computational Linguistics*, 18(1):61–86.

Stuart M. Shieber (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.

Stuart M. Shieber (1993). The problem of logical-form equivalence. *Computational Linguistics*, 19(1):179–190.

Stuart M. Shieber (2007). Probabilistic synchronous tree-adjoining grammars for machine translation: The argument from bilingual dictionaries. In *Proceedings of the HLT-NAACL/AMTA Workshop on Syntax and Structure in Statistical Translation*, pp. 88–95. Rochester, NY.

Stuart M. Shieber and Yves Schabes (1990a). Generation and synchronous tree-adjoining grammars. In *Proceedings of the Fifth International Workshop on Natural Language Generation*, pp. 9–14. Dawson, PA.

Stuart M. Shieber and Yves Schabes (1990b). Synchronous tree-adjoining grammars. In *Proceedings of the Thirteenth International Conference on Computational Linguistics (COLING-1990)*, pp. 253–258. Helsinki, Finland.

Reid Simmons, Dani Goldberg, Adam Goode et al. (2003). GRACE: An autonomous robot for the AAAI robot challenge. *AI Magazine*, 24(2):51–72.

David Smith and Jason Eisner (2006). Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the Workshop on Statistical Machine Translation*, pp. 23–30. New York, NY.

Radu Soricut and Daniel Marcu (2006). Stochastic language generation using WIDL-expressions and its application in machine translation and summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-2006)*, pp. 1105–1112. Sydney, Australia.

Mark Steedman (2000). *The Syntactic Process*. MIT Press, Cambridge, MA.

Ingeborg Steinacker and Ernst Buchberger (1983). Relating syntax and semantics: The syntactico-semantic lexicon of the system VIE-LANG. In *Proceedings of the First Conference of the European Chapter of the Association for Computational Linguistics (EACL-1983)*, pp. 96–100. Pisa, Italy.

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

Andreas Stolcke (2002). SRILM—an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP-2002)*, pp. 901–904. Denver, CO.

Lappoon R. Tang and Raymond J. Mooney (2001). Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning (ECML-2001)*, pp. 466–477. Freiburg, Germany.

Cynthia A. Thompson, Roger Levy and Christopher D. Manning (2003). A generative model for semantic role labeling. pp. 397–408. Cavtat-Dubrovnik, Croatia.

Cynthia A. Thompson and Raymond J. Mooney (1999). Automatic construction of semantic lexicons for learning natural language interfaces. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pp. 487–493. Orlando, FL.

Christoph Tillmann (2003). A projection extension algorithm for statistical machine translation. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP-03)*, pp. 1–8. Sapporo, Japan.

Alan M. Turing (1950). Computing machinery and intelligence. *Mind*, 59:433–460.

Erik Velldal and Stephan Oepen (2005). Maximum entropy models for realization ranking. In *Proceedings of the Tenth Machine Translation Summit (MT Summit X)*. Phuket, Thailand.

Ashish Venugopal, Stephan Vogel and Alex Waibel (2003). Effective phrase translation extraction from alignment models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, pp. 319–326. Sapporo, Japan.

Andrew J. Viterbi (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Stephan Vogel, Hermann Ney and Christoph Tillmann (1996). HMM-based word alignment in statistical translation. In *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING-1996)*, pp. 836–841. Copenhagen, Denmark.

Wolfgang Wahlster, Heinz Marburger, Anthony Jameson and Stephan Busemann (1983). Overanswering yes-no questions: Extended responses in a natural language interface to a vision system. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, pp. 643–646. Karlsruhe, West Germany.

Mengqiu Wang, Noah A. Smith and Teruko Mitamura (2007). What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007)*, pp. 22–32. Prague, Czech Republic.

Ye-Yi Wang and Alex Acero (2003). Combination of CFG and n-gram modeling in semantic grammar learning. In *Proceedings of the 8th European Conference on Speech Communication and Technology (EuroSpeech-2003)*, pp. 2809–2812. Geneva, Switzerland.

Ye-Yi Wang, Li Deng and Alex Acero (2005). Spoken language understanding. *IEEE Signal Processing Magazine*, 22(5):16–31.

David H. D. Warren and Fernando C. N. Pereira (1982). An efficient easily adaptable system for interpreting natural language queries. *American Journal of Computational Linguistics*, 8(3-4):110–122.

David J. Weir (1988). *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw et al. (2007). OntoNotes version 1.0. Linguistic Data Consortium.

Michael White (2004). Reining in CCG chart realization. In *Proceedings of the 3rd International Conference on Natural Language Generation (INLG-2004)*. New Forest, UK.

Michael White and Jason Baldridge (2003). Adapting chart realization to CCG. In *Proceedings of the 9th European Workshop on Natural Language Generation (EWNLG-2003)*. Budapest, Hungary.

Robert W. Wilensky and Yigal Arens (1980). PHRAN—a knowledge-based approach to natural language analysis. Tech. Rep. UCB/ERL M80/34, Electronics Research Laboratory, University of California at Berkeley.

Yorick Wilks (1973). An artificial intelligence approach to machine translation. In Roger C. Schank and Kenneth M. Colby, eds., *Computer Models of Thought and Language*, pp. 114–151. W. H. Freeman & Co., San Francisco, CA.

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

Yuk Wah Wong and Raymond J. Mooney (2007a). Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of North American Chapter of the Association for Computational Linguistics Annual Meeting and Human Language Technology Conference (NAACL-HLT-2007)*, pp. 172–179. Rochester, NY.

Yuk Wah Wong and Raymond J. Mooney (2007b). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual*

Meeting of the Association for Computational Linguistics (ACL-2007), pp. 960–967. Prague, Czech Republic.

William A. Woods, Ronald M. Kaplan and Bonnie Nash-Webber (1972). The lunar sciences natural language information system: Final report. Tech. Rep. 2378, Bolt, Beranek and Newman, Inc., Cambridge, MA.

Dekai Wu (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Dekai Wu (2005). Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pp. 25–30. Ann Arbor, MI.

Dekai Wu and Hongsing Wong (1998). Machine translation with a stochastic grammatical channel. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and COLING-98 (ACL-COLING-98)*, pp. 1408–1415. Montréal, Canada.

Fei Xia and William Lewis (2007). Multilingual structural projection across interlinear text. In *Proceedings of North American Chapter of the Association for Computational Linguistics Annual Meeting and Human Language Technology Conference (NAACL-HLT-2007)*, pp. 452–459. Rochester, NY.

XTAG Research Group (2001). A lexicalized tree adjoining grammar for English. Tech. Rep. IRCS-01-03, IRCS, University of Pennsylvania.

Kenji Yamada and Kevin Knight (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.

Kenji Yamada and Kevin Knight (2002). A decoder for syntax-based MT. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 303–310. Philadelphia, PA.

David Yarowsky and Grace Ngai (2001). Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*, pp. 200–207. Pittsburgh, PA.

Victor Yngve (1962). Random generation of English sentences. In *1961 International Conference on Machine Translation of Languages and Applied Language Analysis*, pp. 66–80. Her Majesty's Stationery Office, London, UK.

John M. Zelle (1995). *Using Inductive Logic Programming to Automate the Construction of Natural Language Parsers*. Ph.D. thesis, Department of Computer Sciences, University of Texas, Austin, TX. Also appears as Artificial Intelligence Laboratory Technical Report AI 96-249.

John M. Zelle and Raymond J. Mooney (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 1050–1055. Portland, OR.

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

Luke S. Zettlemoyer and Michael Collins (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 Computational Natural Language Learning (EMNLP-CoNLL-2007), pp. 678–687. Prague, Czech Republic.

Vita

Yuk Wah "John" Wong was born in Hong Kong in 1979. After finishing high school in 1998, he went to study Computer Science and Information Systems at the University of Hong Kong, where he obtained a Bachelor of Science degree with first class honours in 2001. John is now pursuing his doctorate in Computer Sciences at the University of Texas at Austin. His research interests include natural language understanding and generation, machine translation, and information extraction. Recently, John has been honored with the Best Paper Award at the ACL-2007 conference in Prague for his work on semantic parsing. He will be joining the Google team in Pittsburgh after graduation.

Permanent address: Flat 1702, Kam Ling House, Kam Fung Court,
Ma On Shan, Hong Kong.

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.