The Dissertation Committee for Ruifang Ge
certifies that this is the approved version of the following dissertation:

Learning for Semantic Parsing Using Statistical Syntactic Parsing Techniques

Committee:

Raymond J. Mooney, Supervisor

Jason M. Baldridge

Risto Miikkulainen

Martha Palmer

Bruce W. Porter

Learning for Semantic Parsing Using Statistical Syntactic Parsing Techniques

by

Ruifang Ge, B.S.; M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2010

Dedicated to my loving family.

Acknowledgments

During my dissertation research, many people have generously offered their help and impacted my research. I am sincerely grateful to all of them.

First of all, I would like to thank my advisor, Raymond Mooney, who has been the most important person to my dissertation research. Ray has offered me all the guidance and help that I could ever ask for and more. His broad knowledge, profound observation, and impressive research passion have led me towards better research. I especially appreciate his great patience, understanding, and support during the hard times. Besides, Ray's charming characteristic has made working with him a terrific and memorable experience.

I am thankful to my dissertation committee, Jason Baldridge, Risto Miikkulainen, Martha Palmer, Bruce Porter, and the former member, Ben Kuipers. I have a deeper understanding of the research thanks to their insight and suggestions. I am especially grateful to Jason Baldridge for his valuable advice and help on my job hunting besides research.

I would also like to thank my colleagues in the machine learning group, Sugato Basu, Misha Bilenko, Razvan Bunescu, Tuyen Huynh, Rohit Kate, Prem Melville, Lily Mihalkova, Un Yong Nahm, and Yuk Wah (John) Wong for their friendship and many interesting discussions. Especially I would like to thank Rohit and John for closely collaborating on the semantic parsing project, and Rohit again for helping with many errands when I was not in Austin.

I must also thank Elaine Rich for her encouragement on my research, and the CS staff members Gloria, Katherine, Lydia and Stacy for their administrative

support.

Last but not least, I would like to extend my appreciation to all my friends for their long-term caring and support, and my dear family, my parents, sisters, husband, son, and parents-in-law for their unconditional love.

The research in this thesis was supported by the Defense Advanced Research Projects Agency under grant HR0011-04-1-0007, and NSF under grant IIS-0712097.

RUIFANG GE

The University of Texas at Austin
March 2010

Learning for Semantic Parsing Using Statistical Syntactic Parsing Techniques

Publication No. _____

Ruifang Ge, Ph.D.
The University of Texas at Austin, 2010

Supervisor: Raymond J. Mooney

Natural language understanding is a sub-field of natural language processing, which builds automated systems to understand natural language. It is such an ambitious task that it sometimes is referred to as an AI-complete problem, implying that its difficulty is equivalent to solving the central artificial intelligence problem – making computers as intelligent as people. Despite its complexity, natural language understanding continues to be a fundamental problem in natural language processing in terms of its theoretical and empirical importance.

In recent years, startling progress has been made at different levels of natural language processing tasks, which provides great opportunity for deeper natural language understanding. In this thesis, we focus on the task of semantic parsing, which maps a natural language sentence into a complete, formal meaning representation in a meaning representation language. We present two novel state-of-the-art learned syntax-based semantic parsers using statistical syntactic parsing techniques, motivated by the following two reasons. First, the syntax-based semantic parsing is

theoretically well-founded in computational semantics. Second, adopting a syntax-based approach allows us to directly leverage the enormous progress made in statistical syntactic parsing.

The first semantic parser, SCISSOR, adopts an integrated syntactic-semantic parsing approach, in which a statistical syntactic parser is augmented with semantic parameters to produce a semantically-augmented parse tree (SAPT). This integrated approach allows both syntactic and semantic information to be available during parsing time to obtain an accurate combined syntactic-semantic analysis. The performance of SCISSOR is further improved by using discriminative reranking for incorporating non-local features. The second semantic parser, SYNSEM, exploits an existing syntactic parser to produce disambiguated parse trees that drive the compositional semantic interpretation. This pipeline approach allows semantic parsing to conveniently leverage the most recent progress in statistical syntactic parsing.

We report experimental results on two real applications: an interpreter for coaching instructions in robotic soccer and a natural-language database interface, showing that the improvement of SCISSOR and SYNSEM over other systems is mainly on long sentences, where the knowledge of syntax given in the form of annotated SAPTs or syntactic parses from an existing parser helps semantic composition. SYNSEM also significantly improves results with limited training data, and is shown to be robust to syntactic errors.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Natural language understanding is a sub-field of natural language processing, which builds automated systems to understand natural language. It is such an ambitious task that it sometimes is referred to as an AI-complete problem, implying that its difficulty is equivalent to solving the central artificial intelligence problem – making computers as intelligent as people. Despite its complexity, natural language understanding continues to be a fundamental problem in natural language processing from both a theoretical and empirical standpoint. Theoretically, natural language understanding answers the question of how people interpret language. Empirically, deep natural language understanding is an integral part of natural language interfaces (Androutsopoulos et al., 1995; Zelle and Mooney, 1996; Kuhlmann et al., 2004), question answering (Poon and Domingos, 2009), learning by reading (Barker et al., 2007), and reasoning (Lev et al., 2004).

Due to the availability of a large amount of training data (Marcus et al., 1993; Palmer et al., 2005), advanced statistical methods (Collins, 2004), and massive computing resources, startling progress has been made at different levels of natural language processing tasks which may hopefully finally enable us to reach the final goal of understanding. Much of the research effort has focused on surface level tasks for analyzing structures in which meanings are conveyed, such as part-of-speech tagging (Smith and Eisner, 2005; Goldwater and Griffiths, 2007), chunking (Sang, 2002; Sha and Pereira, 2003) and syntactic parsing (Collins, 1997;

Charniak and Johnson, 2005; Carreras et al., 2008). Significant research has also been pursued on shallow semantic analysis tasks, such as word-sense disambiguation (Ide and Jéronis, 1998; Tanaka et al., 2007), semantic role labeling (Gildea and Palmer, 2002; Carreras and Marquez, 2004, 2005), and information extraction (Califf and Mooney, 1999; Bunescu and Mooney, 2005; Paşca, 2009), which are closer to the goal, by identifying the meanings of target words or finding phrases that fill in the semantic roles of a single predicate or semantic frame.

All this progress provides great opportunity for deeper natural language understanding. In this thesis, we focus on such a task which analyze meanings deep in natural language sentences by leveraging the progress in natural language processing. Specially, we use statistical syntactic parsing techniques for semantic parsing.

1.1 Semantic Parsing

Semantic parsing is the task of mapping a natural language (NL) sentence into a complete, formal *meaning representation* (MR) in a *meaning representation language* (MRL), that is unambiguous which allows for automated reasoning, such as first-order predicate logic. In particular, our research focuses on applications in which the MRL is "executable" and can be directly used by another program to perform some task such as answering questions from a database or controlling the actions of a real or simulated robot. Below is a sample natural language advice given to a simulated soccer player and its formal MR in a coach language (CLang):

If our player 2 has the ball, then position our player 5 in the midfield.

```
((bowner (player our {2}))
 (do (player our {5}) (pos (midfield))))
```

where `bowner`, *ball owner*, is a domain specific predicate in the predefined coach language, which requires a player as its argument. Meaning representations in CLang can be directly understood by a learned simulated player. This differs from representing semantics using more general MRs, such as using the binary predicate "`has(player,ball)`" instead of `bowner` in the example, which is more isomorphic with the syntax, but not directly executable.

The executable MR distinguishes semantic parsing from the related shallow semantic tasks such as semantic role labeling (Carreras and Marquez, 2004) which is quite intentionally designed to be close to the syntax, and not directly executable. From another point view, semantic parsing can also be seen as the task of machine translation which translates an NL sentence into an MR (Shieber and Schabes, 1990; Wong, 2007). Thus it shares much of the challenges in machine translation such as the non-isomorphism between an NL sentence and an MR.

1.2 Approaches

As the standard in computational semantics (Blackburn and Bos, 2005), early semantic parsing systems mainly pursued a hand-built, syntax-based approach (Woods, 1970; Warren and Pereira, 1982; Dowding et al., 1993; Bos et al., 1994), where syntax is used to provide the meaning composition structure in which the meaning of a parent is built from the meanings of its children in the tree. In these hand-built systems, a unification-based grammar is often carefully developed including detailed lexical entries and grammar rules for directing the generation of the correct syntactic structure and unification for semantic composition.

Manually encoding all syntactic and semantic information into the grammar is labor-intensive and brittle. In response to this, a number of learned semantic

parsing approaches were developed, such as Zelle and Mooney (1996) and Miller et al. (1996), and recently Kate et al. (2005), Kate and Mooney (2006), Zettlemoyer and Collins (2005), Wong and Mooney (2006, 2007), and Lu et al. (2008). Most of these approaches departed from the syntax-based approach, but adopted a semantic-driven approach instead, arguing that syntactic parse trees can be more elaborate than needed for meaning composition; the sentence structure needed for meaning composition can be built driven by semantics. On the other hand, Miller et al. (1996) and Zettlemoyer and Collins (2005) were among the few systems which still adopted a syntax-based approach. Miller et al. (1996) learn an integrated syntactic-semantic parsing model from parse trees augmented with semantic information, and Zettlemoyer and Collins (2005) learn a CCG which requires a small set of manually-designed grammar template rules to start with.

1.3 Thesis Contributions

This thesis takes a syntax-driven approach to semantic parsing for the following two reasons:

- The syntax-based semantic parsing is theoretically well-founded in computational semantics: the grammatical relations between phrases or words encoded in a syntactic parse tree allow us to directly read predicate-argument relations in semantics from the tree.

- Adopting a syntax-based approach allows us to directly leverage the enormous progress made in statistical syntactic parsing. Statistical syntactic parsers have improved significantly over the years in terms of quality (Collins, 1999; Charniak, 2000; Charniak and Johnson, 2005; Koo et al., 2008; Carreras et al., 2008) and diversity of grammar formalism (Riezler et al., 2002; Clark and

Curran, 2004). The enormous progress has also boosted progress in the tasks which take syntactic parse trees as input for higher-level interpretation, such as information extraction (Zelenko et al., 2003; Bunescu and Mooney, 2005), information retrieval (Liu et al., 2007) and semantic role labeling (Gildea and Jurafsky, 2002; Carreras and Marquez, 2005).

We introduce two novel learned syntax-based semantic parsers using statistical syntactic parsing techniques. The first parser, SCISSOR (Ge and Mooney, 2005), adopts an integrated syntactic-semantic parsing approach, where a state-of-the-art statistical syntactic parser is augmented with semantics to produce a semantically-augmented parse tree (SAPT); this tree is then translated into a final formal meaning representation. Training requires that sentences be annotated with SAPTs as well as MRs. This integrated approach allows semantic information to be available during parsing time, so that the parser can find a globally most likely parse for both syntactic and semantic interpretation to obtain an accurate combined syntactic-semantic analysis. A discriminative reranking model (Ge and Mooney, 2006) is also developed for incorporating non-local features.

The second parser, SYNSEM (Ge and Mooney, 2009), adopts a pipeline approach for semantic parsing, which exploits an existing syntactic parser to produce disambiguated parse trees that drive the compositional semantic interpretation. With the advancement of statistical syntactic parsing, accurate syntactic parsers are available for many languages and could potentially be used to learn more effective semantic analyzers. Thus, this pipeline approach allows semantic parsing to conveniently leverage the progress in syntactic parsing. This contrasts with Zettlemoyer and Collins (2005) which requires a set of hand-crafted grammar template rules and CCG combinators to start with. Unlike the first approach, it does not require fully-annotated SAPTs for training.

The first approach used the same integrated syntactic-semantic parsing idea to produce augmented parse trees as in Miller et al. (1996), but with significant differences. First, the semantic augmentation in Miller et al. (1996) is designed for domains in which an MR can be represented by a single semantic frame, while the semantic augmentation in our approach is designed for deeply nested MRs which are more expressive. Second, our approach is based on a state-of-the-art syntactic parsing model which is suitable for modeling predicate-argument knowledge in an application domain.

This thesis aims to answer the following three critical questions:

- Can the two proposed learned syntax-based approaches produce accurate semantic parsers?
- What are the strengths and weaknesses of the proposed approaches when compared to each other?
- What are the strengths and weaknesses of the proposed syntax-based approaches when compared to the non-syntax-based approaches? In other words, we want to examine when syntax can help, in the form of annotated SAPTs or syntactic parses from an existing parser.

We show that the two proposed learned syntax-based approaches both produce state-of-the-art performance. The main improvement of SCISSOR and SYNSEM over other systems is on long sentences, due to the prior syntactic knowledge given in the form of annotated SAPTs or syntactic parses from an existing parser that helps semantic composition. When comparing SCISSOR and SYNSEM, we show that SCISSOR outperforms SYNSEM when given sufficient training data, by utilizing the annotated SAPTs. However, when given limited training data, SYNSEM

gives the best results by using an accurate syntactic parser to provide syntactic knowledge.

1.4 Thesis Outline

The outline of the thesis is as follows:

- Chapter 2 introduces syntactic parsing and discriminative reranking on which our semantic parsing algorithms are based. We also provide background knowledge in semantic parsing, including the main application domains and approaches.

- Chapter 3 presents the integrated syntactic-semantic parsing approach called SCISSOR. Specifically, Collins (1997) parsing model 2 is augmented to incorporate the semantic knowledge of an application domain into the model. We report experimental results on two real applications, an interpreter for coaching instructions in robotic soccer and a natural-language database interface. We also take a closer look at the strengths and weaknesses of the parser.

- Chapter 4 presents discriminative reranking for semantic parsing which incorporates non-local syntactic and semantic features. We report experimental results based on SCISSOR in the two real applications.

- Chapter 5 presents the pipeline approach to learning semantic parsers called SYNSEM. Specifically, it uses syntactic parse trees from an existing syntactic parser to drive the interpretation process. The learned parser uses standard compositional semantics to construct alternative MRs for a sentence based on its syntax tree, and then chooses the best MR based on a trained statis-

tical disambiguation model. We report experimental results on the two real applications, and analyze the strengths and weaknesses of the parser.

- Chapter 6 gives future work, and Chapter 7 concludes this thesis.

We note that the material presented in Chapter 3 has appeared in our previous publication Ge and Mooney (2005), the material in Chapter 4 has appeared in Ge and Mooney (2006), and the material in Chapter 5 has appeared in Ge and Mooney (2009).

Chapter 2

Background

In this chapter, we briefly introduce syntactic parsing and discriminative reranking on which our semantic parsing algorithms are based. We also provide background knowledge in semantic parsing, including the main application domains and approaches.

2.1 Statistical Syntactic Parsing

Syntactic parsing is the process of constructing a syntactic parse tree for an input sentence by recursively applying a sequence of context free rewriting rules (see Figure 2.1). The key issue in syntactic parsing is to solve syntactic ambiguity, which arises when a sentence can have more than one syntactic parse tree according to a grammar. For example, *The children ate the cake with a spoon* where the prepositional phrase can be attached to either the noun or the verb.

Statistical parsing models provide a natural way for solving ambiguity by attaching probabilities to each parse tree of a sentence. Probabilistic context free grammars (PCFGs) are one of the most widely used models among them. Formally, a PCFG is a 4-tuple:

$$G = (N, \Sigma, S, R) \tag{2.1}$$

where N is a set of non-terminal symbols, Σ is a set of terminal symbols, and S is a distinguished symbol in N, namely the start symbol. R is a set of rules of the form

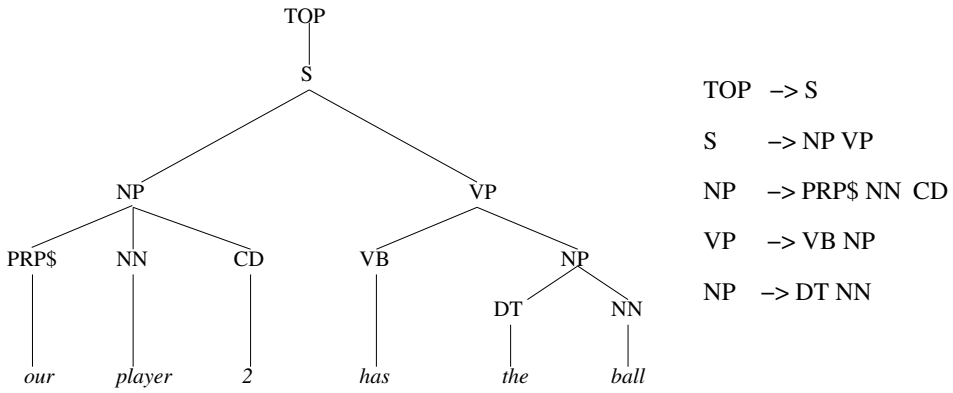


Figure 2.1: A syntactic parse tree and the list of associated rewrite rules.

$LHS \rightarrow RHS$, where $LHS \in N$ and RHS is a sequence of terminals and non-terminals; each rule has a associated probability where the probabilities of all rules are expanding the same non-terminal sum up to one. PCFGs output a parse tree with the highest joint probability $P(T, S)$, where the joint probability is defined as the product of the n applications of the context free rules $LHS_i \rightarrow RHS_i, 1 \leq i \leq n$.

$$\mathcal{P}(T, S) = \prod_{i=1}^{n} \mathcal{P}(LHS_i \rightarrow RHS_i) \tag{2.2}$$

In supervised learning, the probability of each rule is acquired by using maximum likelihood estimation on a set of labeled parse trees through counting with smoothing.

A well-known drawback with PCFGs is their lack of lexicalization – the probabilities of rules are independent of the words involved. For example, in the Penn Treebank (Marcus et al., 1993), the probabilities of the rule VP→V NP with different verbs *take* (32.1%) and *come* (1.1%) are different (Manning and Schütze, 1999), however, they would be the same under PCFGs. Lexicalized PCFGs address this problem by augmenting each non-terminal in a parse tree with its head word, so that the probabilities of the rewriting rules are sensitive to words involved (see

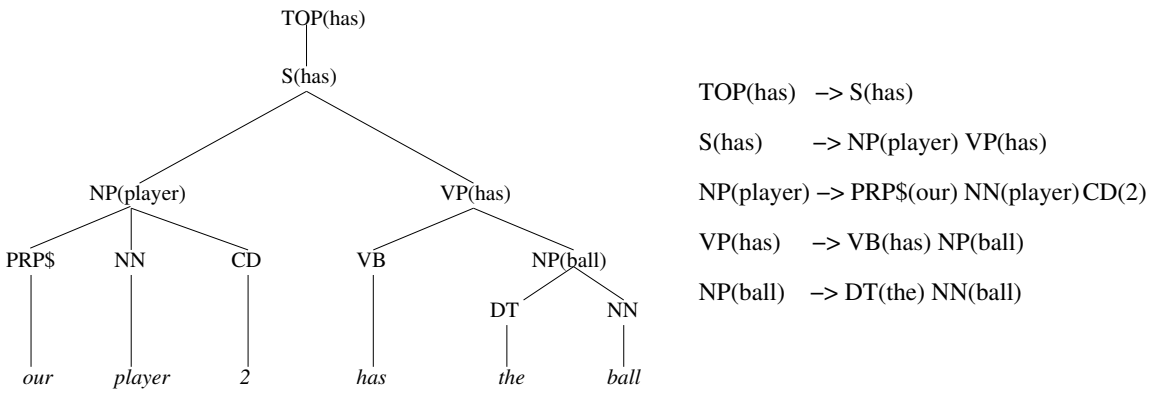


Figure 2.2: A lexicalized parse tree and the list of lexicalized rewrite rules associated.

Figure 2.2). The head is chosen using linguistic rules. For example, the head of a noun phrase is the noun (*player* is the head of the noun phrase *our player 2*).

Lexicalization enormously increases the number of potential rules and makes the direct rule probability estimation infeasible because of sparse data problems. This is particularly true for parsing the Penn Treebank, which is known for its flat tree structures. Collins (2003) points out that there are as many as 12,409 distinct rules from the approximately 40,000 sentences in sections 2-21 of the Penn Treebank. Divide and Conquer strategies are effectively used to tackle this problem (Collins, 1997; Charniak, 1997).

2.1.1 Collins (1997) Parsing Models

In the following part of this subsection, we introduce Collins (1997) parser, one of the best lexicalized statistical parsers. The basic idea of breaking down a rule in Collins (1997) parser is as follows. One child is labeled as a head, and all other children are labeled as modifiers. Expanding the non-terminal in the LHS with its RHS is then broken down into several steps – first generating the head, then

generating the left and right modifiers, respectively, under the assumption that the generation of one modifier is dependent of the head, but independent of other modifiers. By applying the chain rule, the rule probability is calculated as the product of the probabilities associated with the generation steps. Note that sparse data problems are significantly alleviated by relying on the counts of the smaller parts of a rule instead of an entire rule.

While the independence assumption among the modifiers alleviate sparse data problems effectively, it can also lead to incorrect probability estimations. For example, the verb *read* normally only requires one object, thus its probability of taking a second noun phrase as its object should be much lower than the probability of taking the first noun phrase, however, it is not true under the independence assumption.

To capture the dependencies between the modifiers, Collins (1997) builds a series of progressively more complex models that lead to successively improved performance. The first model (CM1) incorporates a distance feature, which is the combination of the distance, intervening words and punctuation between the head and modifier. The second model (CM2) divides the modifiers of a head into complements (essential to the head) and adjuncts (optional to the head). Each head is predicted with a subcategorization frame composed of a set of complements that a head should appear with; and the generation of modifiers is conditioned on the complements in the subcategorization frame that have not yet been fulfilled by the previous modifiers. The third model (CM3) extends the second model to deal with Wh-movement where subcat complements do not appear in their normal place, like in the question *Who did you go out with last night*. In the semantic parsing task, we will use CM2 in parsing sentences because CM2 performs significantly better than CM1, while the most sophisticated model CM3 does not show significant improve-

ment over CM2 (Collins, 1997). Another reason to use CM2 is that the statistics on moved complements required for training CM3 are not labeled in the semantic parsing data.

Below, we formally describe the rule probability estimation in CM2 using the same notation as in Collins (1997). Each non-terminal X in a parse tree is lexicalized with a word, w, and a part-of-speech (POS) tag t. Each rule $LHS \rightarrow RHS$ has the form:

$$P(h) \rightarrow L_n(l_n)...L_1(l_1)H(h)R_1(r_1)...R_m(r_m)$$

where P, H, L, and R are the parent, head child, and left and right children, respectively; each non-terminal is written as $X(x)$, where X is a constituent label, and $x = \langle w, t \rangle$. The rule probability is calculated as the product of the following probabilities:

1. The probability of choosing a head constituent label H: $\mathcal{P}_h(H|P, h)$.

2. The probabilities of choosing the left and right subcategorization frames LC and RC: $\mathcal{P}_{lc}(LC|P, H, h)$ and $\mathcal{P}_{rc}(RC|P, H, h)$.

3. The probabilities of generating the left and right modifiers: $\prod_{i=1..m+1} \mathcal{P}_r(R_i(r_i) |H, P, h, \Delta_{i-1}, RC) \times \prod_{i=1..n+1} \mathcal{P}_l(L_i(l_i)|H, P, h, \Delta_{i-1}, LC)$, where Δ is the distance between the head and modifier, and $L_{n+1}(l_{n+1})$ and $R_{m+1}(r_{m+1})$ are the pseudo non-terminal $STOP$ representing the boundaries of a phrase.

As an example, the probability of the rule VP(has) $\rightarrow$ VB(has) NP(ball) in Figure 2.2 would be estimated as:

$\mathcal{P}_h$(VB|VP,has) $\times$ $\mathcal{P}_{lc}$({}|VP,has) $\times$ $\mathcal{P}_{rc}$({NP}|VP,has) $\times$
$\mathcal{P}_l$(STOP|VP,has,{}) $\times$ $\mathcal{P}_r$(NP(ball)|VP,has,{NP}) $\times$ $\mathcal{P}_r$(STOP|VP,has,{})

In Collins' implementation, a variant of the CKY parser is employed to find a parse tree that maximizes the joint probability of a sentence and its parse tree.

2.2 Discriminative Reranking for Syntactic Parsing

Collins' (1997) parsing models are examples of widely-used history-based parsing models, where a parse tree is represented as a sequence of decisions, and the probability of the tree is then calculated as a product of the probabilities associated with these decisions. For example, in Collins' (1997) parsing models, generating the RHS of a rule is decomposed into a sequence of decisions – first choosing the head, then generating the left and right modifiers; each of these decisions is associated with a probability. While history-based models have many advantages, they can be awkward to incorporate discriminative features, because the choice of features is directly constrained by the choice of the generation decisions. For example, one discriminative feature for predicting correct parse trees that the models have trouble incorporating is different heights of subtree features which can be overlapping (Collins, 2002b).

Ideally, we would like to apply algorithms that incorporate arbitrary discriminative features for directly choosing the best parse tree. In practice, however, such algorithms become infeasible when a large exponential number of candidate trees exist, because there is no feasible way to find the best tree efficiently when arbitrary features are included. Dynamic programming techniques cannot apply in this situation, and the algorithms need to enumerate all parse trees to find the best tree.

Reranking approaches (Collins, 2000; Charniak and Johnson, 2005) address this problem with the additional advantage of both allowing a tree to be represented

using arbitrary features, and also keeping the size of the candidate trees manageable. In such an approach, a baseline model is used to generate a set of top parses only utilizing local features (thus feasible for dynamic programming), and then a second model attempts to rerank the top parses using arbitrary discriminative features as evidence.

Formally, a reranking model for statistical syntactic parsing is composed of three parts (Collins, 2002a): a set of candidate parse trees GEN, which is the top N parse trees of a sentence from a baseline parsing model; a function Φ that maps a sentence x and its parse tree y into a feature vector $\Phi(x, y) \in \mathbb{R}^d$; and a weight vector $\bar{W}$ associated with the set of features. Each feature in a feature vector is a function on a parse tree that maps the tree to a real value. For example, a feature could be the counts of a context-free rule in a parse tree. A special and powerful feature, the score of a parse tree under a baseline model, is often included to take advantage of the baseline model. In reranking models, the parse tree with the highest score under a parameter vector $\bar{W}$ is outputted, where the score is calculated as:

$$score(x, y) = \Phi(x, y) \cdot \bar{W} \tag{2.3}$$

Training a reranking model amounts to estimating the parameter vector $\bar{W}$ using a set of training examples. Popular parameter estimation methods for reranking parse trees include probability models that maximize the likelihood of the training examples, such as maximum entropy models (Collins, 2000). They also include distribution-free methods (Collins, 2004) where the distribution generating the data is unknown, such as the perceptron algorithm (Collins, 2002a), boosting (Collins, 2000), and support vector machines (Joachims, 2002). As an example, we introduce the perceptron algorithm (Rosenblatt, 1958) below, which has proven to be

Inputs: A set of training examples (x_i, y_i^*), $i = 1...n$
Initialization: Set $\bar{W} = 0$
Algorithm:
For $t = 1...T, i = 1...n$
Calculate $y_i = \arg\max_{y \in GEN(x_i)} \Phi(x_i, y) \cdot \bar{W}$
If $(y_i \neq y_i^*)$ then $\bar{W} = \bar{W} + \Phi(x_i, y_i^*) - \Phi(x_i, y_i)$
Output: The parameter vector $\bar{W}$

Figure 2.3: The perceptron training algorithm.

efficient and effective in practical problems while also having the advantage of being extraordinarily simple.

The perceptron training algorithm is shown in Figure 2.3. For each sentence x, one of the candidates y^* that has the highest similarity score with the gold-standard parse tree is chosen as the correct one. In training, all parameters initially are set to 0. The algorithm then goes through the training examples for T iterations, calculating the scores of each candidates using the current parameter vector. In each iteration, for every example, the parse tree with the highest score is chosen. If the best tree is not the correct tree, a simple additive method is used to update the weights of the features which have different values in the two parse trees. Note that the update operation is very efficient – parameter values associated with other features remain unchanged. Collins (2002a) gives a theoretical analysis of the convergence property of this method. If the training data is separable and there is a parameter factor W which makes zero errors on the training data, then the perceptron training algorithm will converge to a parameter vector with zero training error in a finite number of iterations.

The averaged perceptron, a variant of the perceptron algorithm is often used in testing to decrease generalization errors on unseen test examples, where the pa-

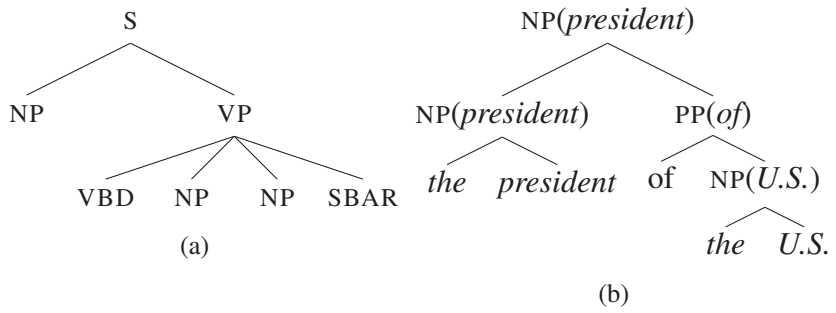


Figure 2.4: The syntactic parse trees for illustrating the reranking features.

rameter vector used in testing is the average of each parameter vector generated during the training process.

2.2.1 Features in Collins and Koo (2005)

In the rest of the section, we briefly introduce the feature types used by Collins (2000) and Collins and Koo (2005) for reranking syntactic parse trees. In Chapter 4, we shall show that the same set of reranking features can be adapted for the task of semantic parsing. The parse trees in Figure 2.4 taken from Collins and Koo (2005) are used for illustration. The head of the rule VP→VBD NP NP SBAR in Figure 2.4(a) is VBD.

1. Rules. These are the counts of unique context-free rules in a syntactic parse. For example, the tree in Figure 2.4(a) has the feature f(VP→ NP NP SBAR)=1.

2. Bigrams. These are the counts of unique bigrams in a constituent. They are also featured with the type of the constituent, and the bigram's relative direction (*left*, *right*) to the head of the constituent. For example, the tree in Figure 2.4(a) has the feature f(NP NP, *right*, VP)=1, where the bigram appears to the right of the head VBD.

3. Grandparent Rules. These are the same as Rules, but also include the non-terminal above a rule. For example, the tree in Figure 2.4(a) has a feature f(VP$\rightarrow$ NP NP SBAR, S)=1, where S is the non-terminal above the rule VP$\rightarrow$ NP NP SBAR.

4. Grandparent Bigrams. These are the same as Bigrams, but also include the non-terminal above the constituent containing the bigram. For example, the tree in Figure 2.4(a) has a feature f(NP NP, *right*, VP, S)=1, where S is the parent of the constituent VP.

5. Lexical Bigrams. These are the same as Bigrams, but also include the lexical heads of the two non-terminals in a bigram.

6. Two-level Rules. These are the same as Rules, but also include the entire rule above a rule, for example, the tree in Figure 2.4(a) has a feature f(VP$\rightarrow$ NP NP SBAR, S$\rightarrow$ NP VP)=1.

7. Two-level Bigrams. These are the same as Bigrams, but also include the entire rule above the constituent containing the bigram. For example, the tree in Figure 2.4(a) has a feature f(NP NP, *right*, VP, S$\rightarrow$ NP VP)=1.

8. Trigrams. These are the counts of unique trigrams in a constituent. This is also featured with the type of the constituent. For example, the tree in Figure 2.4(a) has a feature f(NP NP SBAR, VP)=1, where VP is the type of the constituent containing the trigram.

9. Head-modifiers. These are the counts of unique head-modifier pairs appearing in a constituent, with the types of the constituent and its parent also included. A binary flag *adj* is used to signal if the modifier is adjacent to the head. For example, the tree in Figure 2.4(a) has a feature f(VBD PP, adj=1, VP, S, *left*)=1, where the modifier PP appears directly to the left of the head VBD in the constituent VP under the non-terminals S.

10. PPs. Each feature is the count of a prepositional phrase (PP), the noun phrase (NP) it is attached to, the NP containing it, and the NP it contains with each component lexicalized. For example, the tree in Figure 2.4(b) has a feature f(PP *of*, NP *president*, NP *president*, NP *U.S.*)=1.

11. Distance Head-Modifiers. These features involves the distance between head words.

12. Further Lexicalization. These are the lexicalized version of the previous features except Head-Modifiers, PPs and Distance Head-Modifiers, where all non-terminals are augmented with their lexical heads when the head words are closed-class words.

Recent progress of applying discriminative reranking in syntactic parsing includes (Charniak and Johnson, 2005; Huang, 2008). Besides parsing, discriminative reranking has also been successfully used in a large variety of NLP tasks: POS tagging and chunking (Collins, 2002a), Name Entity recognition (Collins, 2002c), machine translation (Och et al., 2004) and speech recognition (Collins et al., 2005).

2.3 Application Domains for Semantic Parsing

In this section, we introduce three application domains on which semantic parsing has mainly focused, namely, GEOQUERY, ROBOCUP and ATIS.

The first domain we introduce is GEOQUERY, a learned natural language interface to a US geography database. The database has about 800 facts, represented as Prolog assertions. The domain was original chosen for semantic parsing because the quality of semantic parsers can be practically measured by the quality of the final results returned to the user in this domain (Zelle and Mooney, 1996).

The GEOQUERY corpus contains 880 questions paired with their meaning representations in a Prolog-based language, which consists of first-order predicates augmented with several meta-predicates. Below is a sample query with its English gloss:

What are the rivers in Texas?
answer(x_1,(river(x_1),loc(x_1,x_2),equal(x_2,stateid(texas))))

The initial 250 questions were collected by asking undergraduate students to generate English queries for the given database; queries were then manually translated into logical form (Zelle and Mooney, 1996). The corpus was later expanded to 880 questions by collecting more queries from various resources, including queries from real users through a web-based interface to the database (Tang and Mooney, 2001). The initial 250 questions were also translated into Spanish, Japanese and Turkish.

Kate et al. (2005) later developed a functional, variable-free version of the query language called FUNQL, which can be used for semantic parsers not handling logical forms. The MR in FUNQL for the example above is:

```
answer(river(loc_2(stateid('texas'))))
```

where the function `loc_2` binds the second argument of the original predicate `loc`, so `loc_2(stateid('texas'))` denotes the set of entities in the state of Texas; the enclosing function `river` constrains the set to a river subset. The MRs in FUNQL typically demonstrate deeply-nested structures. Table 2.1 gives the statistics on the corpus.

The second domain we introduce is the ROBOCUP domain. ROBOCUP (`www.robocup.org`) is an international AI research initiative using robotic soccer as its primary domain. In the Coach Competition, teams of agents compete on

	CLANG	GEOQUERY	
		FUNQL	LOGICAL MRL
MRG Nonterminals	12	13	14
MRG Productions	134	133	50
Ave. NL Length	22.52	7.57	7.57
No. Unique Words	337	280	280

Table 2.1: Statistics on the CLANG and GEOQUERY corpora.

a simulated soccer field and receive advice from a team coach in a formal language called CLANG. In CLANG, tactics and behaviors are expressed in terms of if-then rules. Below is a sample rule with its English gloss:

If our player 2 has the ball, then position our player 5 in the midfield.

```
((bowner (player our {2}))
 (do (player our {5}) (pos (midfield))))
```

The CLANG corpus (Kate et al., 2005) contains 300 pieces of coaching advice, randomly selected from the log files of the 2003 ROBOCUP Coach competition. Each formal instruction was translated into English by one of four annotators. Table 2.1 gives the statistics on the corpus.

The CLANG MRL is sometimes not isomorphic to the semantics of an NL. In the above example, in the MR, the predicate `pos` only takes one argument `midfield`; while in the NL, the corresponding word *position* takes both the player and *midfield* as arguments.

The last application domain we introduce is the Air Travel Information Services (ATIS) domain (Price, 1990), which is an ARPA-sponsored benchmark for speech recognition and understanding. The ATIS corpus is a collection of spoken questions about air travel, their written form and meaning representations in the

SQL database query language. A sample database query, paired with its SQL query is given below:

Show me the flights from Boston.
```
SELECT flight_id FROM flight WHERE from_airport = 'boston'
```

Some queries in the corpus are context dependent, which need to be interpretated within a discourse. The corpus shows interesting language phenomena in speech language such as flexible word order and the deletion of words.

The semantic parsing task in ATIS can often be simplified to filling a single semantic frame, where the structure among these fillers become less important. For example, the semantic frame associated with the example used above is:

$$\begin{bmatrix} \text{AIR-TRANSPORTATION} & \\ \text{SHOW} & \text{FLIGHT} \\ \text{ORIGIN} & \begin{bmatrix} \text{CITY} & \textit{Boston} \end{bmatrix} \end{bmatrix}$$

2.4 Semantic Parsing Approaches

In this section, we review the approaches developed for semantic parsing. Early semantic parsing systems mainly built hand-crafted grammars for specific application domains, which can be brittle and hard to be ported to other domains. Thus researchers started to investigate various learned approaches which are more robust and easily portable. Section 2.4.1 introduces the early hand-built syntactic-based systems, and Sections 2.4.2 to 2.4.4 introduce the machine learning approaches.

2.4.1 Hand-built Syntax-based Approaches

The syntax-based semantic parsing approaches are based on compositional semantics in which the meaning of a complex expression is determined by the meanings of its parts, and the structure in which those parts are combined. As standard in computational semantics (Blackburn and Bos, 2005), syntax is used to provide the meaning composition structure. Each node in a syntactic parse tree has an associated semantics; the analysis of semantics is driven by the structure of a parse tree, where the meaning of a parent is built from the meanings of its children in the tree.

LUNAR (Woods, 1970), CHAT80 (Warren and Pereira, 1982), GEMINI (Dowding et al., 1993), VERBMOBIL parser (Bos et al., 1994), LKB (Copestake and Flickinger, 2000), and OPENCCG (White and Baldridge, 2003; Baldridge et al., 2007) are among the most notable hand-built semantic parsers. In these approaches, a unification-based grammar is often developed including domain-specific lexical entries and grammar rules. The lexical entries contain a large amount of lexically specific information specifying selectional restrictions. For example, an entry for the word *walks* can specify that syntactically, it requires a subject to be a single and 3rd-person noun; semantically, it requires its subject to be able to walk. The grammar rules direct the correct unification for semantic composition. For example, a head-subject phrase rule can specify that when combining a non-head child and a head child, the non-head child should fulfill the head child's subject if selectional restrictions are met.

PRECISE (Popescu et al., 2003, 2004) is a syntax-based semantic parser specially designed for building NL interfaces to databases. It requires a hand-built lexicon to relate words to semantic concepts and a set of semantic constraints to define correct meaning composition. According to the lexicon and semantic con-

straints, a class of queries is defined as a group of semantically tractable questions, which have a unique semantic parse. They showed that over 90% of the context-independent questions in ATIS are semantically tractable, while only 80% of the queries in GEOQUERY are semantically tractable, which suggests that GEOQUERY is a harder problem for semantic parsing. The hand-crafted lexicon and semantic constraints are also used to correct syntactic errors in a syntactic parse.

2.4.2 Learned Syntax-based Approaches

Hand-built syntax-based approaches require intensive knowledge engineering, and can be brittle and not easily portable. In response to this, learned syntax-based semantic parsing approaches have been developed.

Miller et al. (1994, 1996) augment nodes in a syntactic parse tree with semantic labels which represent semantic concepts in ATIS, and train a statistical hidden understanding parsing model to find the best augmented parse tree. The tree is then converted into a non-recursive semantic frame using a probabilistic semantic interpretation model. Training the model requires fully-annotated augmented parse trees. Figure 2.5 shows a sample augmented parse tree similar to that in (Miller et al., 1996).

In a more recent work, Miller et al. (2000) apply their approach to information extraction, but using an improved head-driven parsing model. The model is similar to that of Collins (1997), but the decomposed parameters are not as elegant as that in Collins parser, without modeling subcategorization and structural preference. Further, each nonterminal label in the model is the combination of a syntactic and semantic label. Without careful smoothing, its parameter estimation is potentially subject to much greater sparse-data problems.

Zettlemoyer and Collins (2005) learn a semantic parser based on combi-

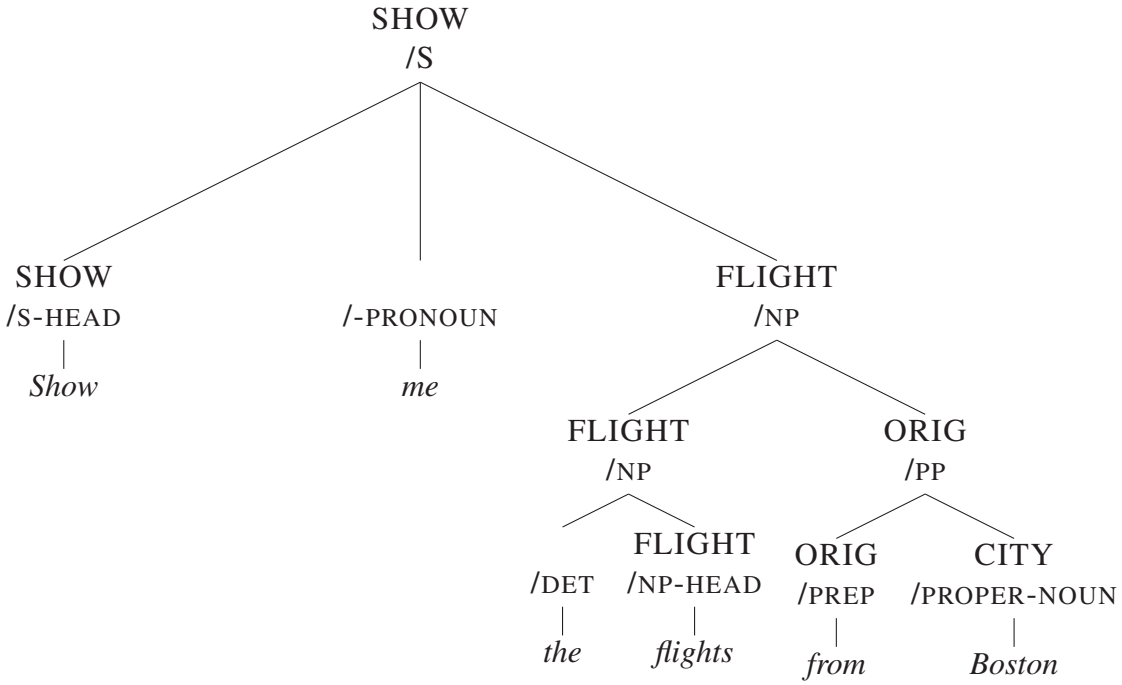


Figure 2.5: A sample augmented parse tree in the ATIS domain, similar to that in (Miller et al., 1996).

natory categorial grammar (CCG) (Steedman, 2000), which is trained directly on NL sentences paired with their meaning representations. The approach does not require fully-annotated augmented parse trees or a CCG, but requires a small set of carefully-designed grammar templates to start with, which specify possible syntactic categories and semantic functions (*output category*) for each type of predicates (*input trigger*) in a meaning representation language. For example, a constant predicate c requiring no argument can trigger the following template rule:

Input trigger:	any constant c
Output category:	NP : c

Initially, the output category is assigned to all words in an example, thus the initial CCG contains many spurious lexical items. A log-linear model is then learned to prune away the spurious items . This work was later improved by Zettlemoyer and Collins (2007) to relax CCG to handle relatively flexible word order and the deletion of words, and in Zettlemoyer and Collins (2009) to allow context-dependent semantic parsing. To handle several types of non-isomorphism between syntax and semantic representations, such as in Figure 5.6, where in syntax, a function (P_POS) takes other function (P_DO)'s semantic argument (P_PLAYER) as argument, it might require using larger lexical categories.

In Chapters 3 and 5, we introduce two semantic parsing algorithms, SCISSOR and SYNSEM, which learn syntax-based semantic parsers. The SCISSOR approach is similar to that in (Miller et al., 1996, 2000), but augments the Collins parsing model, which is state-of-the-art and suitable for modeling predicate-argument knowledge in an application domain (see Chapter 3). Besides, the combined syntactic-semantic nonterminals in SAPTs are carefully smoothed in SCISSOR. The SYNSEM approach is based on an existing syntactic parser which can easily leverage progress

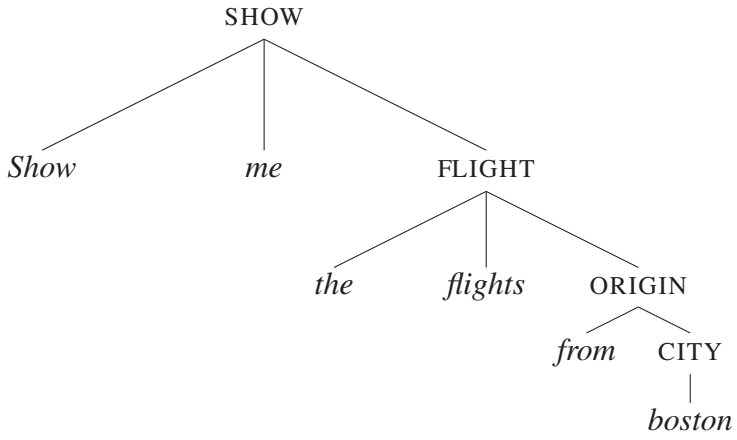


Figure 2.6: A semantic parse tree for the sentence in Figure 2.5.

in syntactic parsing. Unlike SCISSOR, it does not require fully-annotated SAPTs for training.

2.4.3 Learned Semantic Grammars

Another major approach to semantic parsing is semantic grammars (Hendrix et al., 1978), where nonterminals are purely semantic labels, and words can appear in production rules directly. Semantic grammars embed more lexical information (especially about heads of phrases) directly into the syntax rules, which can greatly increase the efficiency of the parsing. The parse trees for meaning composition can also be more concise than the ones generated by syntax-based approaches. For example, in Figure 2.5, since *the* conveys no meaning, the noun phrase *the flights* actually has the same meaning as its child *flights*, thus no meaning composition happens in this phrase. Figure 2.6 shows a more concise semantic parse tree generated by a semantic grammar.

In this section, we review several approaches which learn directly from NL

sentences paired with their meaning representations. Thus, they require less supervision than the approaches in Section 2.4.2. Typically, for each semantic concept in an application domain, a set of production rules are learned where the left-hand-side (LHS) of a production rule is a nonterminal for the concept, and the right-hand-side (RHS) is a string of terminals for the associated context words, and nonterminals for all of its argument concepts.

SILT (Kate et al., 2005) induces a semantic grammar using a bottom-up rule-learning method. For each semantic concept, the examples are first labeled as positive or negative according to the concept's appearance in the meaning representations; then production rules for that concept are learned from these examples using a rule-learning method. The rule-learning method learns both a string version (SILT-STRING) of the production rules which is generalized from the NL sentences directly, and a tree version (SILT-TREE) which is generalized from the syntactic parses of the NLs. Since the concepts are in nested structures, where one concept can have other concepts as its arguments, the production rules are learned in a bottom-up manner: the productions for a concept are only learned after its argument concepts are learned. The parsing is deterministic and lacks the robustness of a statistical model.

WASP (Wong and Mooney, 2006, 2007) learns a synchronous context-free grammar (SCFG) for semantic parsing based on machine translation techniques, where an SCFG can be seen as a synchronous pair of a meaning representation grammar and a semantic grammar. The steps of learning an SCFG are as follows. First, the NL sentence and meaning representation pairs in the training corpus are aligned by a statistical word alignment model. Then, semantic grammar rules are inferred from the alignments in a bottom-up manner. Finally, a maximum entropy model is used to estimate the probability of these rules.

Since various ways can be employed in expressing a semantic concept, the production rules learned in SILT and WASP can fail to capture some context using direct pattern matching. To address this problem, Kate and Mooney (2006) introduce a semantic parser called KRISP based on string kernels. For each semantic concept in a meaning representation language, a string-kernel-based classifier is trained to capture a potentially infinite number of production rules to form a kernel-based semantic grammar. KRISP can also make use of syntactic parse trees as prior knowledge by utilizing a tree-kernel instead (Kate, 2007).

The RHS of a production rule learned in the previous approaches include all of a concept's arguments and its associated context, which can be sparse. Therefore, LU (Lu et al., 2008) proposes a generative parsing model, where the generation of the RHS of a production rule is decomposed into several decision steps as a Markov process, inspired by Collins (1997) head-driven parsing models. It also has a reranking model for utilizing non-local features not available in the base model.

2.4.4 Other Learned Approaches

CHILL (Zelle and Mooney, 1996) is one of the earliest learning approaches to semantic parsing, which learns a deterministic shift-reduce parser using inductive logic programming (ILP) (Muggleton, 1992). Control rules are induced to decide the parsing actions given the context, such as when to introduce a semantic concept, and when to fill an argument. The algorithm learns directly from NL sentences paired with their meaning representations. It requires a lexicon to relate words to semantic concepts, which can be acquired using lexicon learning methods (Thompson and Mooney, 2003). Tang and Mooney (2001) revised CHILL to produce COCKTAIL by utilizing multiple clause constructors to obtain more expressive power.

He and Young (2005) treat semantic parsing as a tagging problem, where each word is assigned a label that encodes the structure information in a semantic parse tree. More specifically, the label of a word is a vector of all semantic labels on the path, from the word to the root of a semantic parse. As an example, the word *from* in Figure 2.6 would have a label ⟨ORIGIN,FLIGHT,SHOW⟩. A drawback of this encoding is the sparse data problem. For example, if we have already seen the sentence *the flight from Boston to Austin*, and know that *from* represents an attribute ORIGIN, we should be able to analyze the meaning of *from* in *the train from Boston to Austin* correctly. However, since these two *from*s would have different labels (FLIGHT.ORIGIN and TRAIN.ORIG) in the system, the information from the previous example will not be helpful. An HMM like model is trained directly on NL sentences labeled with their MRs.

For simple semantic parsing tasks, such as booking the flights in the ATIS domain, the semantic parsing task can be simplified to filling a single semantic frame, where the structure among these fillers become less important. CHANEL (Kuhn and De Mori, 1995) adopts a decision tree approach, where each SQL attribute has a corresponding decision tree to decide if an attribute should be included in the query. Macherey et al. (2001) use a phrase-based machine translation approach to translate a sentence into the list of attributes in a semantic frame.

Chapter 3

Semantic Parsing Integrating Syntax and Semantics

This chapter presents a learning semantic parser, which we call SCISSOR, short for *Semantic Composition that Integrates Syntax and Semantics to get Optimal Representations*. It first uses an integrated statistical parser to produce a *semantically-augmented parse tree* (SAPT), in which each non-terminal node has both a syntactic and semantic label. A compositional-semantics procedure is then used to map the augmented parse tree into a final meaning representation. We evaluate the system in CLANG and GEOQUERY (Section 2.3).

3.1 Motivation

As mentioned in Chapter 1, statistical syntactic parsers have improved significantly over the years (Collins, 1999; Charniak, 2000; Clark and Curran, 2004; Charniak and Johnson, 2005; Miyao and Tsujii, 2005; Koo et al., 2008; Carreras et al., 2008). The enormous progress has also boosted progress in the tasks of computational semantics which take syntactic parse trees as input for semantic interpretation, ranging from interpreting shallow semantics like information extraction (Bunescu and Mooney, 2005) and semantic role labeling (Gildea and Jurafsky, 2002; Pradhan et al., 2005a), to interpreting deep meaning representation like semantic parsing (Lev et al., 2004; Curran et al., 2007).

However, syntactic parses directly from a syntactic parser are sometimes

suboptimal for semantic interpretations since no semantic information is utilized during parsing. A syntactic parser may fail to rank the syntactic parse which is the best for semantic interpretations the highest, or it may generate syntactic errors which would have been corrected with semantic information available.

A promising approach to this problem is to integrate syntactic and semantic interpretation into a single statistical model, instead of using a pipeline process. This allows semantic information to be available during parsing time, so that the parser can find a globally most likely parse for both syntactic and semantic interpretation to obtain an accurate combined syntactic-semantic analysis (Miller et al., 1994, 1996, 2000). In semantic parsing[1], Miller et al. (1994, 1996) augment nodes in a syntactic parse tree with semantic labels which represent semantic concepts in ATIS, and train a statistical hidden understanding parsing model to find the best augmented parse tree. The tree is then converted into a non-recursive semantic frame using a probabilistic semantic interpretation model. In information extraction, Miller et al. (2000) apply their approach for semantic parsing, by using an improved head-driven parsing model.

In this chapter, we adopt the integrated syntactic-semantic parsing approach for semantic parsing which handles deeply nested meaning representations. Among the state-of-the-art syntactic parsing models, we choose to augment Collins (1997) parsing model 2. A major reason is that it is suitable for incorporating semantic knowledge of an application domain. Concretely, Collins (1997) parsing model 2 models head-nonhead dependencies, which mimic the underlying predicate-argument dependencies; it also models subcategorization, a set of complements that a head should appear with, which mimic the underlying semantic subcategorization, a set

[1]Most ATIS queries are in fact conceptually very simple, so meaning representation in this work often amounts to a single semantic frame (see Section2.3)

of arguments that a predicate should appear with. An additional minor reason for choosing this model is the availability of Bikel (2004)'s implementation of Collins' parser, which is designed to be easily extensible.

The augmented statistical parsing model first generates a semantically augmented parse tree (SAPT), in which each internal node includes both a syntactic and semantic label. Once a SAPT is generated, an additional step is required to translate it into a final formal meaning representation in a meaning representation language. Training requires sentences annotated with both gold-standard SAPTs and MRs. We present experimental results on corpora for both CLANG and GEOQUERY demonstrating that SCISSOR performs better than other systems on CLANG, and competitive to other systems using the same MRL on GEOQUERY. Analysis on CLANG shows that SCISSOR works especially well on long sentences, where syntax is crucial for meaning composition.

The remainder of the Chapter is organized as follows. Section 3.2 describes how semantic knowledge is represented in the integrated syntactic-semantic parsing algorithm. Section 3.3 introduces the basic framework of SCISSOR, followed by Section 3.4 which elaborates on corpus annotation and meaning composition from SAPTs. After that, Section 3.5 formally describes the augmented syntactic-semantic parsing model, and Section 3.6 presents the experimental results.

3.2 Representing semantic knowledge as a Meaning Representation Language Grammar

SCISSOR assumes that semantic knowledge of an application domain is encoded in an unambiguous meaning representation language grammar (MRLG), which specifies the set of predicates in the domain and the semantic constraints on the predicates' arguments. Specifically, in an MRLG, the left-hand side (LHS)

If our player 2 has the ball, then position our player 5 in the midfield.

```
((bowner (player our {2}))
 (do (player our {5}) (pos (midfield))))
```

Figure 3.1: A sample NL and MR pair in CLANG.

PRODUCTION	PREDICATE
RULE→(CONDITION DIRECTIVE)	P_RULE
CONDITION→(bowner PLAYER)	P_BOWNER
PLAYER→(player TEAM {UNUM})	P_PLAYER
TEAM→our	P_OUR
UNUM→2	P_UNUM
DIRECTIVE→(do PLAYER ACTION)	P_DO
ACTION→(pos REGION)	P_POS
REGION→(midfield)	P_MIDFIELD

Table 3.1: Sample production rules for parsing the CLANG example in Figure 3.1 and their corresponding predicates .

of a production rule is a nonterminal, and the right-hand side (RHS) is a string of terminals and non-terminals. Each production rule introduces a single *predicate* in the MRL, where the *type* of the predicate is given by the nonterminal in the LHS, and the number and types of its arguments are defined by the nonterminals in the RHS. The MR of a predicate is the RHS of its production rule, with nonterminals replaced by real arguments.

Given an MRLG, a meaning representation (MR) can be uniquely parsed, a standard requirement for computer languages. At the same time, an MR parse gives a predicate-argument structure in the application domain. Figure 3.1 shows a sample instruction in CLANG, and Figure 3.2 (a) and (b) show the condition part's MR parse and predicate-argument structure using the MRLG in Wong (2007). Sample MRLG productions for parsing this example and their associated predicates are shown in Table 3.1, where the predicate P_PLAYER takes two arguments (a_1 and a_2)

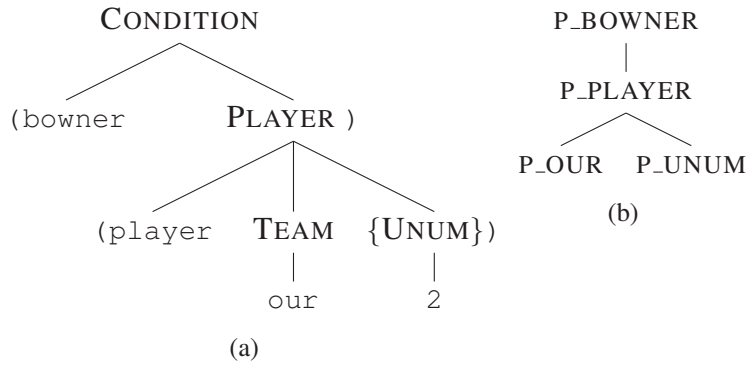


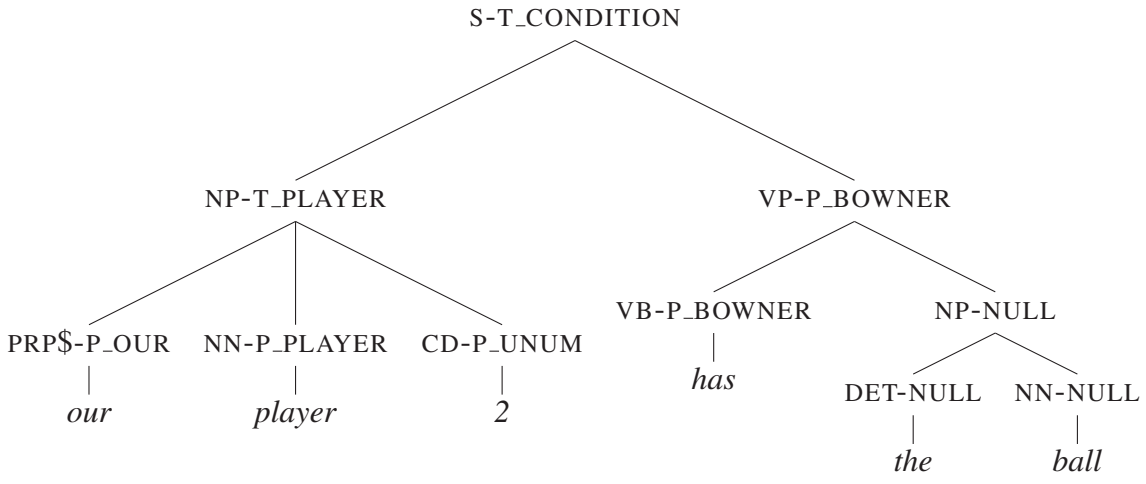
Figure 3.2: (a) The MR parse and (b) predicate-argument structure for the condition part of the CLANG in Figure 3.1 (the nodes for parentheses are not separately shown for brevity).

of type TEAM and UNUM (uniform number). Since a predicate-argument structure uniquely determines an MR string, in the following discussion, we sometimes use predicate-argument structure for MR.

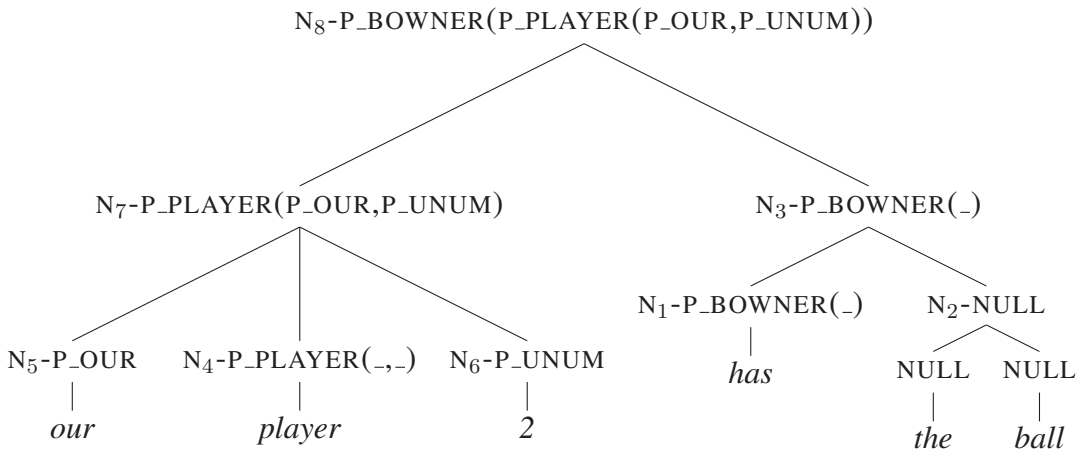
3.3 Semantic Parsing Framework

This section describes the basic framework for our integrated syntactic-semantic parsing algorithm. First, a statistical parser integrating syntactic and semantic information is used to construct a SAPT that captures the semantic interpretation of individual words and the basic predicate-argument structure of a sentence. Next, a recursive procedure is used to compositionally construct an MR for each node in a SAPT from the semantic label of the node and the MRs of its children.

In a SAPT, each node in the parse tree is annotated with a semantic label. Figure 3.3(a) shows the SAPT for the condition part of the CLANG example in Figure 3.1. The semantic labels which are shown after dashes are predicates and types (types are augmented with the prefix T_ for clarity) in CLANG. The semantic



(a) SAPT



(b) Semantic Derivation

Figure 3.3: (a) The SAPT and (b) semantic derivation for the condition part of the example in Fig. 3.1.

Function:COMPOSEMR(N, G)
Input: The root node N of a SAPT;
an MRLG, G.
Notation: X_{MR} is the MR of node X.
Output: N_{MR}

$C_i :=$ the ith child node of $N, 1 \leq i \leq n$
$C_h =$ GETSEMANTICHEAD(N) // see Section 3.3
$C_{h_{MR}} =$ COMPOSEMR(C_h, G)
$N_{MR} = C_{h_{MR}}$
for each other child C_i where $i \neq h$
$C_{i_{MR}} =$ COMPOSEMR(C_i, G)
ADDARGUMENT$(N_{MR}, C_{i_{MR}}, G)$ // see Section 3.3
return N_{MR}

Figure 3.4: Computing an MR from a SAPT.

label on an internal node is the child predicate which takes other child predicates as arguments (*head predicate*); when the predicate has all arguments filled (completed), its type is used as the node's semantic label for better generalization in the tree representation (see elaboration in Section 3.4). For example, P_PLAYER is the head predicate of the NP node covering *our player 2*, since it takes other child predicates as arguments (P_PLAYER(P_OUR,P_UNUM)); its type PLAYER is used as the semantic label of the parent node, since it is completed at the node. A special semantic label NULL is used for nodes that do not correspond to any predicate in the domain.

Figure 3.4 shows the basic algorithm for composing an MR from a SAPT. Figure 3.3(b) shows the *semantic derivation* for constructing an MR from the SAPT in Figure 3.3(a) using this algorithm. Nodes are numbered in the order in which the construction of their MRs are completed. The first step, GETSEMANTICHEAD, determines which of a node's children is its *semantic head* based on having a matching

semantic label (has an equal predicate or belongs to the type). In the example, node N_3 is determined to be the semantic head of the sentence, since its semantic label, P_BOWNER, matches N_8's semantic label. Next, the MR of the semantic head is constructed recursively. The semantic head of N_3 is clearly N1. Since N_1 is a part-of-speech (POS) node, its semantic label directly determines its MR, which becomes P_BOWNER(_). Once the MR for the head is constructed, the MR of all other non-head children are computed recursively, and ADDARGUMENT assigns their MRs to fill the arguments in the head's MR to construct the complete MR for the node. Argument constraints are used to determine the appropriate filler for each argument. Since, N_2 has a NULL label, the MR of N_3 also becomes P_BOWNER(_). When computing the MR for N_7, N_4 is determined to be the head with the MR: P_PLAYER(_,_). ADDARGUMENT then assigns N_5's MR to fill the TEAM argument and N_6's MR to fill the UNUM argument to construct N_7's complete MR: P_PLAYER(P_OUR,P_UNUM). This MR in turn is composed with the MR for N_3 to yield the final MR for the sentence: P_BOWNER(P_PLAYER(P_OUR,P_UNUM)).

The precise meaning composition algorithm depends on the methodology of corpus annotation. We give more details in the next section together with corpus annotation.

3.4 Corpus Annotation and Meaning Composition

This section describes how SAPTs for training SCISSOR were manually annotated to encode semantic knowledge in the training corpus. Since the annotation methodology directly decides meaning composition from SAPTs, meaning composition is also described here. The SAPT annotation on CLANG took one annotator about two weeks' time, and the annotation on GEOQUERY took one week.

To get an initial syntactic parse tree for each sentence in the training corpus, Collins (1997) parsing model 2 (Bikel, 2004) was trained on all sections (00-24) of the WSJ corpus of Penn Treebank (Marcus et al., 1993). The trees produced for the corpus were then manually corrected.

Given a correct syntactic parse tree, semantic annotation starts with adding semantic labels to individual words, called *semantic tags*, to the POS nodes in a SAPT. If a predicate is conveyed by a single word (e.g. *player* for P_PLAYER in Figure 3.3(a)), then this word is labeled with the predicate. If a predicate is conveyed by a phrase (e.g. *has the ball* for P_BOWNER), then only one word is labeled with the predicate, where the syntactic head word (Collins, 1997) is preferred; all other words in the phrase will be used to provide context for determining the word's semantic label during parsing. These unlabeled words in phrases and words conveying no meaning (e.g. *the*) are labeled with the tag NULL. Meaning representation of a word is simply its predicate with arguments unfilled (e.g. P_PLAYER(_,_) for the word *player*).

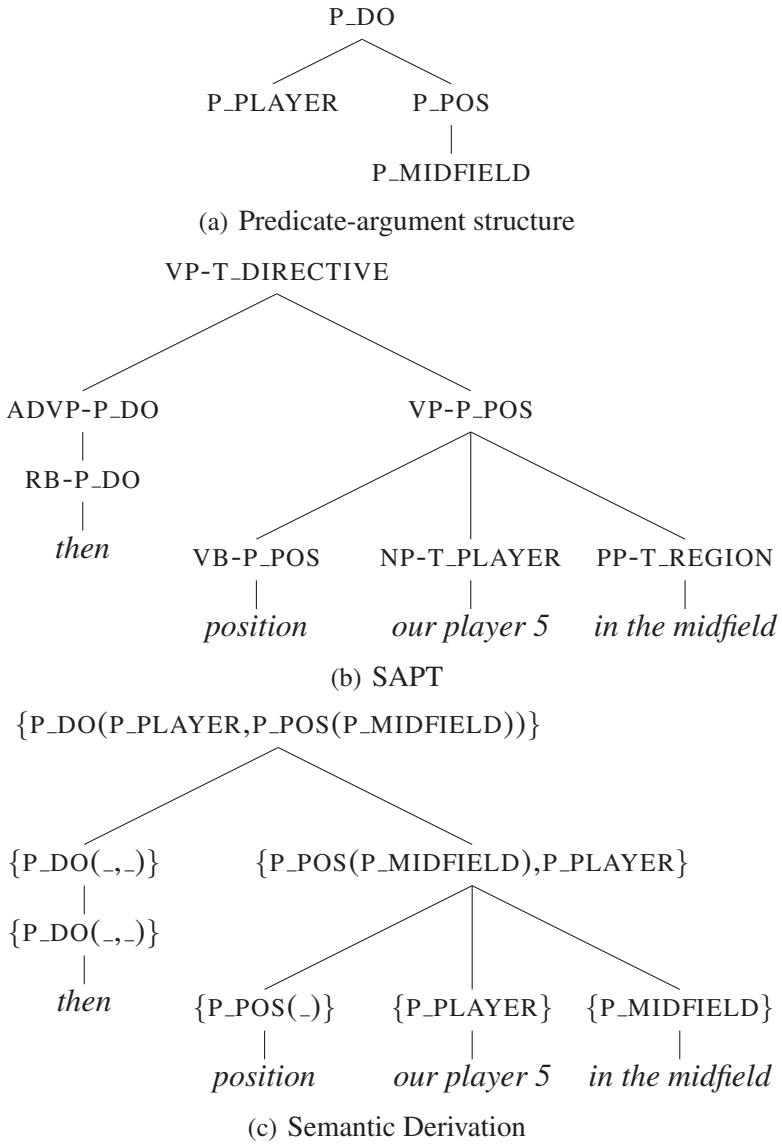
After that, semantic labels are added to the internal nodes in a SAPT in a bottom-up manner. For each node, one of its children is chosen as the semantic head, from which it inherits a semantic label, where the semantic head is chosen to be the child whose predicate takes other child predicates as arguments in the MR; if the semantic head's predicate becomes completed at the parent node, then the predicate's type is used as the the parent node's label (e.g. T_CONDITION). Using the type instead of the specific predicate as the semantic label when a predicate is completed ensures better generalization. It is also similar to the X-bar schema used for head projection in its syntactic counterpart part, e.g., a verb projects upwards to a VP when all its syntactic complements are found. In Figure 3.3(a), the root node inherits the semantic label from its semantic head VP-P_BOWNER, which

takes NP-P_PLAYER as an argument in the MR; since the predicate P_BOWNER is completed at the parent node, its type T_CONDITION is used as the root's semantic label. The meaning representation of an internal node can be composed using the COMPOSEMR process in Figure 3.4.

Three cases require special handling in semantic annotation and meaning composition. First, while many nodes successfully inherit semantic labels from semantic heads whose predicates take all other children's predicates as arguments, many other nodes fail to find such semantic heads. This is due to non-isomorphism in the syntactic parse and the MR parse, where a syntactic node may represent multiple *disjoint* predicates not in predicate-argument relations in the MR. For example, in Figure 3.5, the lower VP node has three child predicates P_POS, P_PLAYER (T_PLAYER) and P_MIDFIELD (T_REGION), where in the syntactic parse, both P_PLAYER and P_MIDFIELD are attached to P_POS, while in the semantic parse, P_POS takes P_MIDFIELD but not P_PLAYER as an argument in the MR. Thus the node represents two disjoint predicates P_POS and P_PLAYER.

Ideally, to add semantic labels to such syntactic nodes representing multiple disjoint predicates, we could use the combination of the disjoint predicates for high discriminative power. However, it would most likely worsen the data sparsity problem since the labels on SAPTs are already a combination of syntactic and semantic labels. Therefore, one of the disjoint predicates is chosen as the semantic label of a node (revised *semantic head*) where the predicate on the syntactic head is preferred (P_POS).

In this case, the basic meaning composition algorithm COMPOSEMR in Figure 3.4 needs to be extended, since the MR of a node is no longer a single predicate, and cannot be composed by simply attaching the non-head children's MRs to the semantic head's MR. In the extension, a node's MR is represented by an ordered



(a) Predicate-argument structure

(b) SAPT

(c) Semantic Derivation

Figure 3.5: (a) The predicate-argument structure, (b) SAPT, and (c) semantic derivation for the directive part of the example in Figure 3.1 (The internal structure of P_PLAYER in (a) and multi-word leaf nodes in (b) and (c) are omitted for brevity).

list of disjoint predicates, and is composed of two steps. First, the children's MR lists are merged into a single list. When merging the lists, the semantic head's list is merged first, followed by other children's lists ordered from left to right. The merging order ensures that the predicates on a semantic head receive high composition priority in the next step. Next, for each predicate in the merged list, we go through the list to find its missing arguments, attach them to the predicate, and remove them from the list. If an attached argument has missing arguments by itself, they are also found. The yield is the MR of the parent node. We note that this process is deterministic and only allows one interpretation for a SAPT: once an argument is attached to a predicate in the front of a list, predicates in the back of the list who may also require it cannot have it.

Consider the meaning composition for the directive part in Figure 3.1 using the extended algorithm (see Figure 3.5(c)). To compose an MR for the lower VP node covering *position our player 5 in the midfield*, the first step merges the children's MR lists {P_POS(_)}, {P_PLAYER}, and {P_MIDFIELD} into a new list {P_POS(_), P_PLAYER, P_MIDFIELD}, where P_POS is at the front since it matches the parent node's semantic label. The second step finds arguments for each predicate in the ordered list, thus P_POS has its argument P_MIDFIELD filled, and the list becomes {P_POS(P_MIDFIELD), P_PLAYER}, the MR of the parent node. To compose an MR for the root node, the first step merges the children's MR lists into a new list {P_DO(_,_), P_POS(P_MIDFIELD), P_PLAYER}, where P_DO is at the front. The second step finds P_DO's arguments P_POS and P_PLAYER, and the list becomes {p_do(P_PLAYER,P_POS(P_MIDFIELD))}, which is the MR of the directive part.

Second, in order for COMPOSEMR to be able to construct the MR for a node, given an argument predicate of certain type, the production rule for a head predicate must identify a unique argument for the argument predicate to fill. How-

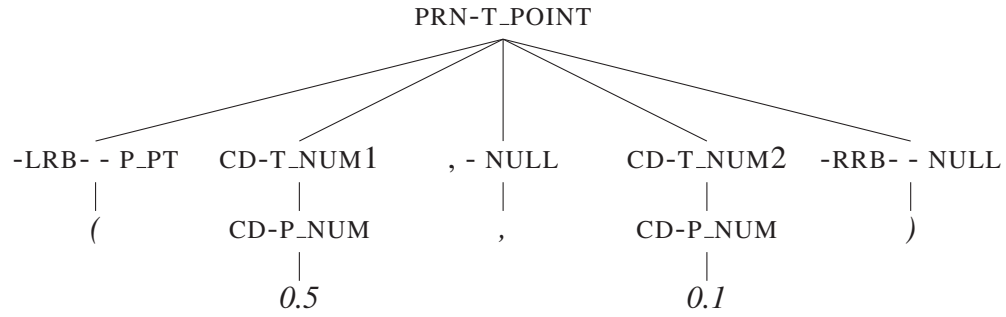


Figure 3.6: Adding new type labels to disambiguate arguments.

ever, some predicates take multiple arguments of the same type, thus the algorithm would fail to decide a unique argument. For example, the predicate P_PT (POINT→(pt NUM NUM)) representing an xy-coordinate has two arguments of the same type NUM. In this case, extra nodes are inserted in the tree with new introduced types which are only used to specify the argument. In Figure 3.6, the new introduced type T_NUM1 specifies that the predicate P_NUM (NUM→ 0.5) below of type NUM should be the first argument of the head predicate P_PT.

Third, the meaning composition algorithm fails to compose a complete MR when a predicate in an MR is not represented by any word. For example, in CLANG, *our player* is frequently just referred to as *player* and the predicate P_OUR must be inferred (*default value*), because advice is given from a team coach's perspective. Therefore, the meaning composition algorithm also learns from the training corpus what arguments of a predicate can have default value, and uses them when needed. We note that traditionally, it is the task of pragmatic processing, not semantic processing. It works for CLANG because there are limited entities that it is reasonable for them to have default values that can be learned. In a more wide-coverage application, a separate pragmatic processing is needed.

3.5 Integrated Syntactic-Semantic Parsing Model

In this section, we formally introduce the integrated syntactic-semantic parsing model based on Collins (1997) head-driven parsing model 2. We first briefly review Collins' parsing models necessary for understanding this chapter (see detailed description in Section 2.1.1). After that, we describe the integrated model with semantic output added. The augmented model is more complex than the origional one and requires careful smoothing, thus, Section 3.5.2 describes smoothing in detail. Last, Section 3.5.3 gives implementation details.

Recall that Collins (1997) parsing model 2 (see Section 2.1.1) is a generative lexicalized PCFG model. In a generative PCFG, a syntactic parse tree is seen as expanding nonterminals using production rules in a CFG grammar recursively starting with a start nonterminal, and the probability of a tree is the product of all probabilities associated with the production rules used. However, probabilities in a PCFG are insensitive to lexical information and unable to provide adequate discriminative power. In addition, the model also suffers from the sparse-data problem since many production rules may only occur very few times. Therefore, Collins (1997) proposes a sequence of progressively improved head-driven PCFG models where the structure preference of a syntactic head is modeled, as used in many lexicalized syntactic formalisms such as CCG (Steedman and Baldridge, to appear) and HPSG (Pollard and Sag, 1994). In model 2, each syntactic label in a syntactic parse tree is lexicalized with a head word and its POS tag, and the expansion of a nonterminal (LHS) using the RHS of a production rule is decomposed into a decision sequence: first generating the head, then generating the subcategorizations of the head modeling structural preference, and lastly generating the left and right modifiers constrained on both the head and its subcategorizations. The resulting model is more compact and discriminative. We choose to augment model 2 instead of the

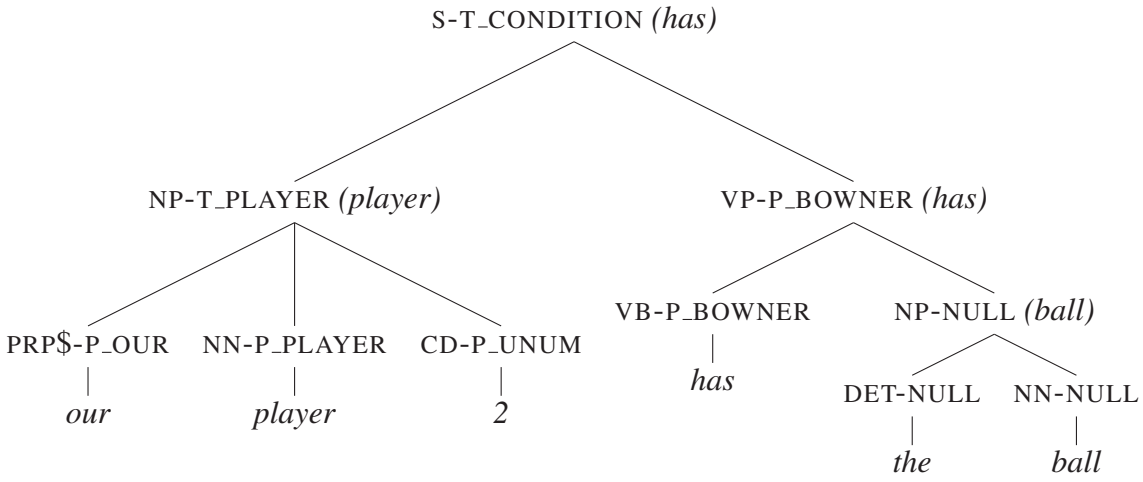


Figure 3.7: The lexicalized SAPT for the SAPT in Figure 3.3(a).

more sophisticated model 3, because model 3 does not show significant improvement over model 2. It is also because the moved complement information necessary for training model 3 is not labeled in our training corpora.

3.5.1 Integrating Semantics into the Model

In our integrated parsing model for generating SAPTs, each syntactic component in Collins' parsing model is augmented with its semantic counterpart: a semantic tag for a POS tag, a semantic label for a syntactic label, and a semantic subcategorization for a syntactic subcategorization, where a semantic tag is the semantic label attached to a head word, and a left/right semantic subcategorization is the set of semantic labels appearing on the left/right nodes of a head node, introduced for modeling a predicate's argument requirements which are used when generating semantic labels of modifiers. Figure 3.7 shows a lexicalized SAPT (omitting POS and semantic tags lexicalized to nonterminals), where the root node

S-T_CONDITION has the head child VP-P_BOWNER with left semantic subcategorization {T_PLAYER} and empty right semantic subcategorization.

Formally, in the augmented syntactic-semantic parsing model, the expansion of a nonterminal (LHS) using the RHS of a production rule

$$P(h) \rightarrow L_n(l_n)...L_1(l_1)H(h)R_1(r_1)...R_m(r_m)$$

is decomposed into the same decision sequence as in Collins' parsing model 2, but with augmented labels:

1. Generating a head label H with probability $\mathrm{Pr}_h(H|P,h)$.
2. Generating the left and right subcategorization frames LC and RC with probabilities $\mathrm{Pr}_{lc}(LC|P,H,h)$ and $\mathrm{Pr}_{rc}(RC|P,H,h)$.
3. Generating the left and right modifiers with probabilities $\mathrm{Pr}_l(L_i,l_i|P,H,h,\Delta_{i-1},LC)$ and $\mathrm{Pr}_r(R_i,r_i|P,H,h,\Delta_{i-1},RC)$.

where the uppercase letters stand for nonterminal labels, and lowercase letters stand for lexicalized heads; the letter P (p) stands for parent, H (h) for head, L (l) for left, and R (r) for right (other symbols are described in Section 2.1.1). Each augmented nonterminal X is in the form of $\langle X_{syn}, X_{sem}\rangle$, where the subscript syn refers to the syntactic part, and sem refers to the semantic part; each augmented lexicalized head x is in the form of $\langle w, t_{syn}, t_{sem}\rangle$, where w is a head word, t_{syn} is a POS tag, and t_{sem} is a semantic tag; the augmented left and right subcategorization frames LC and RC are in the form of $\langle LC_{syn}, LC_{sem}\rangle$ and $\langle RC_{syn}, RC_{sem}\rangle$.

As an example, the probability of expanding the root node in Figure 3.7 using the decomposed steps (omitting the distance measure and the complement symbol -C attached to NP as in Collins' parser) is calculated as:

$$
\begin{aligned}
&\Pr_h(\text{VP-P\_BOWNER} \mid \text{S-T\_CONDITION}, \textit{has}) \\
&\times \Pr_{lc}(\{\{\text{NP}\},\{\text{T\_PLAYER}\}\} \mid \text{S-T\_CONDITION}, \text{VP-P\_BOWNER}, \textit{has}) \\
&\times \Pr_{rc}(\{\{\},\{\}\} \mid \text{S-T\_CONDITION}, \text{VP-P\_BOWNER}, \textit{has}) \\
&\times \Pr_l(\text{NP-T\_PLAYER}(\textit{player}) \mid \text{S-T\_CONDITION}, \text{VP-P\_BOWNER}, \textit{has}, \{\{\text{NP}\},\{\text{T\_PLAYER}\}\}) \\
&\times \Pr_l(\text{STOP} \mid \text{S-T\_CONDITION}, \text{VP-P\_BOWNER}, \textit{has}, \{\{\},\{\}\}) \\
&\times \Pr_r(\text{STOP} \mid \text{S-T\_CONDITION}, \text{VP-P\_BOWNER}, \textit{has}, \{\{\},\{\}\})
\end{aligned}
$$

where STOP is a special symbol specifying the boundary of a constituent. The semantic subcategorizations decide that there is a T_PLAYER to the head's left, and no semantic labels required to the right. After the child NP-T_PLAYER is generated, T_PLAYER is removed from the left subcategorization.

3.5.2 Smoothing

The integrated parsing model allows semantic information to be available during parsing time, so that the parser can find a globally most likely parse for both syntactic and semantic interpretation. However, since the model is more complex than the original model, it has higher risk of sparse data problem. In this section, we discuss how the parameters (probabilities associated with the generation steps) are further decomposed and smoothed in the integrated model.

The parameters are first decomposed using the chain rule where syntactic features are generated first, followed by semantic features conditioned on syntactic features (only the parameters for generating the left modifiers are shown):

$$
\begin{aligned}
\Pr_h(H|C) &= \Pr_{h_{syn}}(H_{syn}|C) \times \Pr_{h_{sem}}(H_{sem}|C, H_{syn}) \\
\Pr_{lc}(LC|C) &= \Pr_{lc_{syn}}(LC_{syn}|C) \times \Pr_{lc_{sem}}(LC_{sem}|C, LC_{syn}) \\
\Pr_l(L_i(l_i)|C) &= \Pr_{l_{syn}}(L_{i_{syn}}(lt_{i_{syn}}, lw_i)|C) \times \Pr_{l_{sem}}(L_{i_{sem}}(lt_{i_{sem}}, lw_i)|C, L_{i_{syn}}(lt_{i_{syn}}))
\end{aligned}
$$

where C represents the context on which each parameter is conditioned, and lw_i, $lt_{i_{syn}}$, and $lt_{i_{sem}}$ are the head word, POS tag, and semantic tag associated with the non-terminal L_i. Words are generated independently in both syntactic and semantic outputs.

The model is then further simplified by making the independence assumption to condition syntactic output only on syntactic features, and semantic output only on semantic features:

$$\begin{aligned}
\Pr_h(H|C) &= \Pr_{h_{syn}}(H_{syn}|C) \times \Pr_{h_{sem}}(H_{sem}|C, H_{syn}) \\
&= \Pr_{h_{syn}}(H_{syn}|C_{syn}) \times \Pr_{h_{sem}}(H_{sem}|C_{sem}) \\
\Pr_{lc}(LC|C) &= \Pr_{lc_{syn}}(LC_{syn}|C) \times \Pr_{lc_{sem}}(LC_{sem}|C, LC_{syn}) \\
&= \Pr_{lc_{syn}}(LC_{syn}|C_{syn}) \times \Pr_{lc_{sem}}(LC_{sem}|C_{sem}) \\
\Pr_l(L_i(l_i)|C) &= \Pr_{l_{syn}}(L_{i_{syn}}(lt_{i_{syn}}, lw_i)|C) \times \Pr_{l_{sem}}(L_{i_{sem}}(lt_{i_{sem}}, lw_i)|C, L_{i_{syn}}(lt_{i_{syn}})) \\
&= \Pr_{l_{syn}}(L_{i_{syn}}(lt_{i_{syn}}, lw_i)|C_{syn}) \times \Pr_{l_{sem}}(L_{i_{sem}}(lt_{i_{sem}}, lw_i)|C_{sem})
\end{aligned}$$

Note that the syntactic and semantic parameters are still integrated in the model to find the globally most likely parse. We have also tried different ways of conditioning syntactic output on semantic features and vice versa, but they failed to show significant improvement. Our explanation is that the integrated syntactic and semantic parameters have already captured the benefit of this integrated approach in our experimental domains.

The syntactic parameters are decomposed and smoothed as in Collins (1997), and the semantic parameters are decomposed and smoothed as follows. Since the semantic parameters do not depend on syntactic features under the independence assumption, the subscript sem can be safely omitted. The parameter $\Pr_{l_{sem}}(L_i(lt_i, lw_i)$ $|P, H, w, t, \Delta, LC)$ for generating a left modifier is again decomposed as:

BACK-OFF LEVEL	$\Pr_h(H\|...)$	$\Pr_{lc}(LC\|...)$	$\Pr_{l1}(L_i\|...)$	$\Pr_{l2}(lt_i\|...)$	$\Pr_{l3}(lw_i\|...)$
1	P,w,t	P,H,w,t	P,H,w,t,Δ,LC	P,H,w,t,Δ,LC, L_i	P,H,w,t,Δ,LC, L_i, lt_i
2	P,t	P,H,t	P,H,t,Δ,LC	P,H,t,Δ,LC, L_i	P,H,t,Δ,LC, L_i, lt_i
3	P	P,H	P,H,Δ,LC	P,H,Δ,LC, L_i	L_i, lt_i
4	–	–	–	L_i	lt_i

Table 3.2: Conditioning features for each back-off level in semantic parameters.

$$\begin{aligned} & \Pr_{l1}(L_i|P,H,w,t,\Delta,LC) \\ \times\ & \Pr_{l2}(lt_i|P,H,w,t,\Delta,LC,L_i) \\ \times\ & \Pr_{l3}(lw_i|P,H,w,t,\Delta,LC,L_i(lt_i)) \end{aligned}$$

where the parameters are the probabilities for generating the semantic label, semantic tag, and head word of a left modifier respectively. We point out that the smoothing is different from its syntactic counterpart, where the generation of a syntactic label and POS tag pair $L_i(lt_i)$ is not decomposed into two parameters as in $\Pr_{l1}$ and $\Pr_{l2}$. This is because semantic tags are essentially more specific than syntactic tags, and require more smoothing. Table 3.2 shows the back-off levels for each semantic parameter. The probabilities from these back-off levels are interpolated using the techniques in Collins (1997).

3.5.3 Implementation details

As in Collins' parsing model 2, the integrated parsing model does not rely on an external POS or semantic tagger. Instead, it uses the following method to provide candidate tags for parsing. It classifies words into known and unknown words, where unknown words are those occurring less than 3 times in the training data, and words in the test data that were not seen in training. Note that the unknown word threshold is smaller than the one in Collins (1997) since the training corpora for semantic parsing are small. For known words, the candidate tags are those that

have been seen with the word in the training data; for unknown words, the candidate POS tags are those that have been seen with any unknown words in the training data, and the candidate semantic tags are limited to those that have been seen with the word's associated POS tag during training.

In training, counts needed for estimating a model are directly collected from a training corpus. In testing, a CKY-style algorithm used in Collins (1997) is extended to find the best SAPT that maximizes the joint probability of a sentence and SAPT. The beam width[2] (Bikel, 2004) is set to 10^4. After that, the meaning composition algorithm is used to compose an MR, and return the MR of the root node as the MR of an example . An example fails to *return* an MR when the root node's MR has more than one disjoint predicate, or a predicate is incomplete.

3.6 Experimental Evaluation

3.6.1 Methodology

We experimented with SCISSOR on both the CLANG and GEOQUERY corpora (Section 2.3). Since SCISSOR does not handle MRLs with logical variables, in GEOQUERY, we use FUNQL as the MRL. We also give results on the small GEOQUERY corpus containing 250 examples, GEO250. Detailed information on the corpora were shown in Section 2.3.

SCISSOR was evaluated using standard 10-fold cross validation. We measured the number of test sentences that returned MRs, and the number of MRs that were correct. For CLANG, an MR is correct if it exactly matches the correct MR, up to reordering of the arguments of commutative operators like `and`. For GEO-

[2]In beam search, each cell in the chart is reduced by discarding all items whose probabilities are lower than $\frac{1}{\beta}$ times the probability of the best item in the cell. This β is called the *beam width*.

QUERY, an MR is correct if it retrieves the same answer as the gold-standard query, thereby reflecting the quality of the final result returned to the user. Since even a single mistake in an MR could totally change the meaning of an example, no partial credit was given for examples with partially-correct SAPTs. The performance of the parser was then measured in terms of *precision*, *recall*, and *F-measure*:

$$\text{Precision} = \frac{\text{No. of correct MRs}}{\text{No. of test sentences returning MRs}} \tag{3.1}$$

$$\text{Recall} = \frac{\text{No. of correct MRs}}{\text{No. of test sentences}} \tag{3.2}$$

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision + Recall}} \tag{3.3}$$

We compared SCISSOR with the systems briefly described below (also see Section 2.4). There are great variances among these systems, in terms of MRLs used, and syntactic knowledge and engineering features required. Two MRLs are available for GEOQUERY, among which FUNQL produces more deeply nested MR parses than PROLOG, so systems using FUNQL can have greater performance loss due to non-isomorphism between syntax and semantic representations than systems using PROLOG (Wong and Mooney, 2007). All systems require NL sentences paired with their MRs for training. Besides, extra supervision and engineering features are used in some systems and not available to other systems, which may also affect performance. For example, SCISSOR requires the annotation of SAPTs, COCKTAIL uses a hand-built lexicon for GEOQUERY, and Z&C requires hand-built CCG lexical entries and templates. Below are the systems, and their variances are summarized in Table 3.8(a) and Table 3.8(b):

- COCKTAIL (Tang and Mooney, 2001) is a deterministic shift-reduce parser based on inductive logic programming. It requires a lexicon to start with:

	CLANG	GEOQUERY	
		FUNQL	LOGICAL MRL
COCKTAIL (2001)	√	–	√
SCISSOR (2005)	√	√	–
WASP (2006)	√	√	–
KRISP (2006)	√	√	–
λ-WASP (2007)	–	–	√
Z&C (2007)	–	–	√
LU (2008)	√	√	–

(a) Corpus

	SAPT annotation	Hand-built components	Reranking
COCKTAIL (2001)	–	√	–
SCISSOR (2005)	√	–	–
WASP (2006)	–	–	–
KRISP (2006)	–	–	–
λ-WASP (2007)	–	–	–
Z&C (2007)	–	√	–
LU (2008)	–	–	√

(b) Prior knowledge and reranking

Figure 3.8: (a) Corpora and MRLs, and (b) syntactic knowedge and reranking used in the systems, ordered by publication time.

a *hand-built* lexicon for GEOQUERY, and a learned lexicon using WOLFIE (Thompson and Mooney, 1999) for CLANG. It provides results on both CLANG and GEOQUERY: on CLANG, it fails to handle training sets larger than 160 examples due to intensive memory requirements.

- WASP (Wong and Mooney, 2006) and λ-WASP (Wong and Mooney, 2007). WASP is a semantic parser based on machine translation techniques using synchronous context-free grammars (SCFG). In GEOQUERY, it uses FUNQL as the MRL since it cannot handle languages with logical variables. The work λ-WASP is an extension of WASP for handing logical forms, and is tested on the PROLOG-based MRL of GEOQUERY.

- KRISP (Kate and Mooney, 2006) is a semantic parser based on string kernels. It also cannot handle logical forms and uses FUNQL on GEOQUERY.

- Z&C (Zettlemoyer and Collins, 2005, 2007) is a probabilistic semantic parser using CCG, where Zettlemoyer and Collins (2007) is an improvement over Zettlemoyer and Collins (2005) to allow more flexibility in CCG grammars. It requires hand-built CCG templates and lexical items for function words (e.g. *what*) as prior knowledge. In GEOQUERY, we compared our results to the result with the highest F-measure in Zettlemoyer and Collins (2007), which only uses one split of data into training and test sets containing 600 and 280 examples respectively. No results have been reported on CLANG. To make it work on CLANG, the hand-built components and CCG combinators may need to be revised. To handle non-isomorphism between an NL and its MR, such as in Figure 3.5 (also see Section 5.3), larger lexical categories may need to be introduced.

	CLANG			GEOQUERY		
	Precision	Recall	F-measure	Precision	Recall	F-measure
COCKTAIL	–	–	–	89.9	79.4	84.3
SCISSOR	**89.5**	**73.7**	**80.8**	92.1	72.3	81.0
WASP	88.9	61.9	73.0	87.2	74.8	80.5
KRISP	85.2	61.9	71.7	93.3	71.7	81.1
λ-WASP	–	–	–	92.0	**86.6**	**89.2**
Z&C	–	–	–	**95.5**	83.2	88.9
LU	82.5	67.7	74.4	89.3	81.5	85.2

Table 3.3: Performance of semantic parsers on CLANG and GEOQUERY.

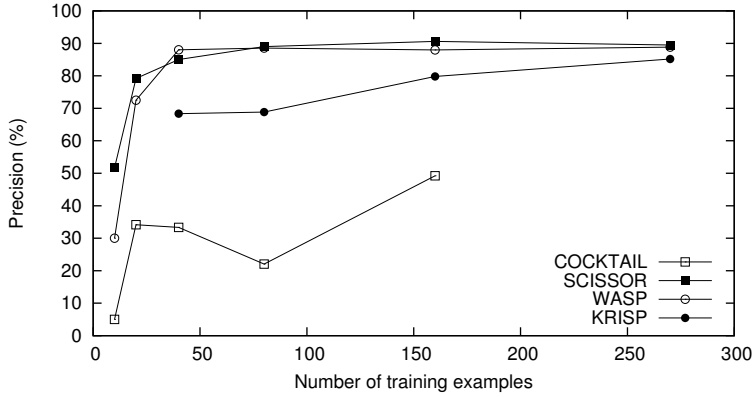
- LU (Lu et al., 2008) is a generative parsing model using semantic grammar, which also has a reranking model for utilizing non-local features not available in the base model. It uses FUNQL on GEOQUERY.

3.6.2 Results

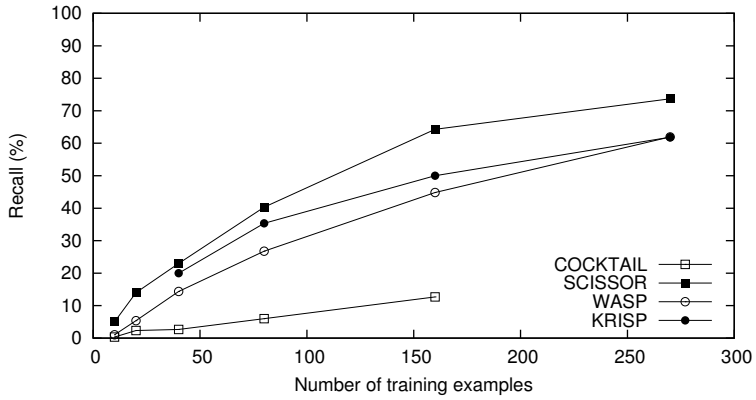
Performance of semantic parsers on CLANG and GEOQUERY is shown in Table 3.3[3], and learning curves for the available systems on CLANG and GEOQUERY are shown in Figure 3.9 and Figure 3.10. Several observations can be made:

- In CLANG, SCISSOR substantially outperformed all other systems.
- In GEOQUERY, the semantic parsers using the PROLOG MRL (COCKTAIL, λ-WASP and Z&C) outperformed the semantic parsers using FUNQL (SCISSOR, WASP, KRISP and LU) due to the deeply nested MR structure in FUNQL.
- In GEOQUERY, SCISSOR performed highly competitively all across the learning curves to KRISP and WASP which also used FUNQL. However, its perfor-

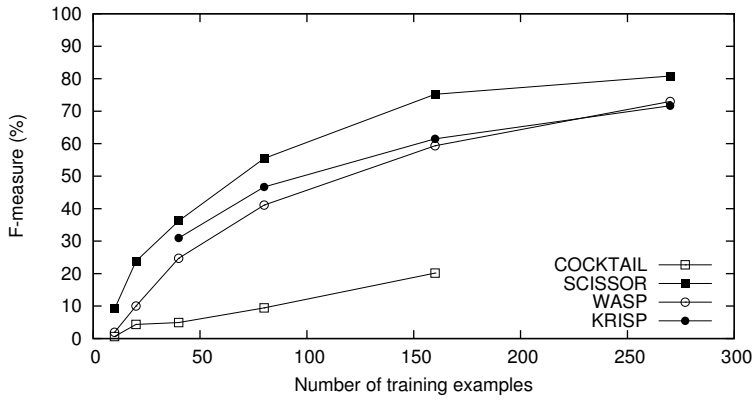
[3]Before reranking, Lu et al. (2008)'s F-measure is 67.8% on CLANG, and 84.0% on GEOQUERY



(a) Precision

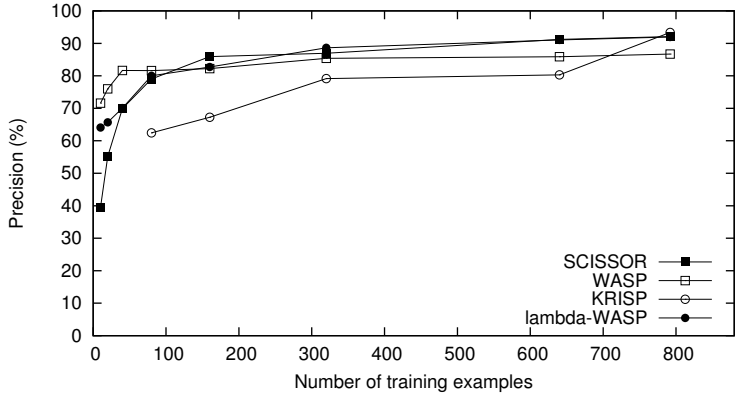


(b) Recall

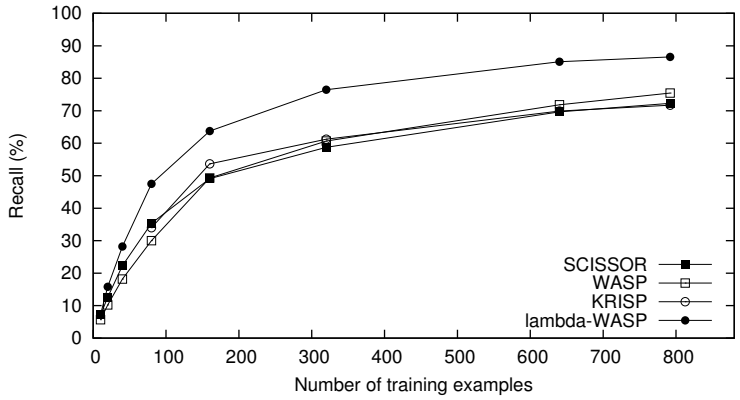


(c) F-measure

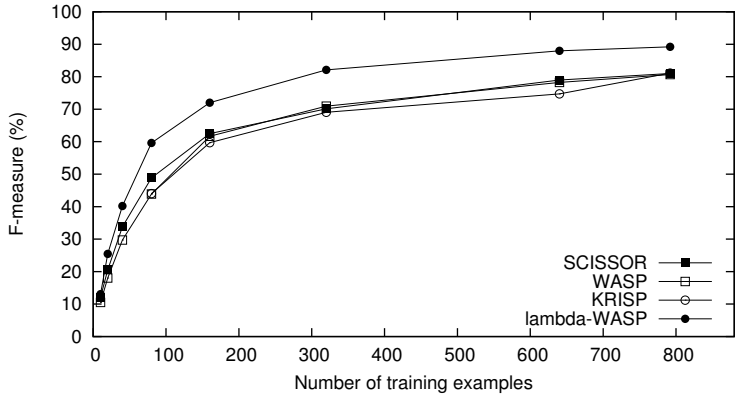
Figure 3.9: Learning curves for semantic parsers on CLANG.



(a) Precision



(b) Recall

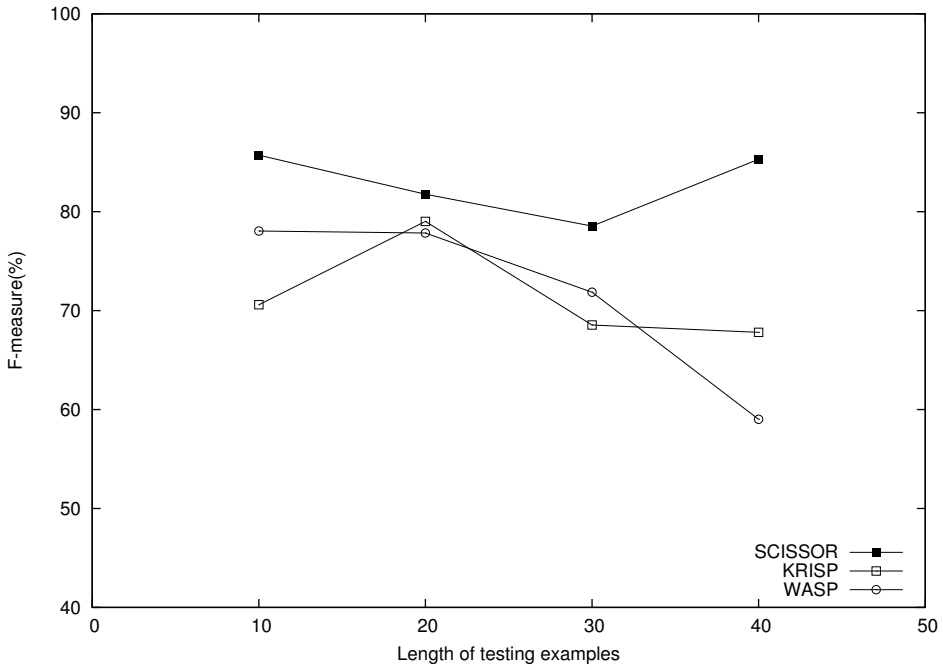


(c) F-measure

Figure 3.10: Learning curves for semantic parsers on GEOQUERY.

mance using all training data is lower than that of LU. Although the reranking model used in LU helps, its performance before reranking is still higher than SCISSOR (F-meaure $= 84.0$). This suggests that when sentences are short, the sentence structure used for meaning composition can be captured well by a semantic grammar learned directly from sentences paired with their MRs. In fact, the approaches utilizing non-syntactic prior knowledge (LU, WASP, and KRISP) can be sometimes even more flexible in exploring optional feasible sentence structures for meaning composition. We shall give detailed analysis in Section 5.7.2.2 after introducing SYNSEM which also utilizes the knowledge of syntax (given by an existing syntactic parser). We note that the improvement of ZU over WASP and KRISP on GEOQUERY is due to the decomposed model it used which is more robust for learning semantic grammars on short sentences (Section 2.4.3); However, when sentences are long, its performance is worse than that of CLANG and KRISP which do not decompose a production rule, as illustrated on CLANG before reranking (F-meaure $= 67.8$).

Figure 3.11 gives the detailed look at the F-measures on sentences within different length ranges on CLANG (range 41-50 not shown since only 5 sentences fall into this category), where the sentence count for each length range is shown in Figure 3.11 (b), and most sentences are within the ranges of $11-20$ and $21-30$. It shows that the great improvement of SCISSOR over WASP and KRISP appeared when sentences are long: while WASP and KRISP degraded significantly as sentences became longer, SCISSOR did not show such degradation. It suggests that utilizing syntactic knowledge learned from the SAPT annotation in SCISSOR successfully guides the correct compositional semantic analysis; which can be hard for the semantic-grammar-based parsers to learn when sentences are long. We note that



(a) Results

NL length	0-10	11-20	21-30	31-40	41-50
Sentence count	22	98	137	38	5
Corresponding point in the figure above	*10*	*20*	*30*	*40*	–

(b) NL-length distribution

Figure 3.11: CLANG results on test sentences within length ranges.

	GEOQUERY		
	Precision	Recall	F-measure
COCKTAIL	80.9	**79.2**	80.0
SCISSOR	**98.5**	74.4	**84.8**
WASP	95.4	70.0	80.8
KRISP	91.3	71.6	77.5
λ-WASP	91.8	75.6	82.9
LU	91.5	72.8	81.1

Table 3.4: Performance of semantic parsers on GEO250

.

KRISP performed better on longer sentences than WASP because of the string kernel it utilized to learn production rules which can capture richer syntactic variation (Section 2.4.3).

Performance of available semantic parsers on GEO250 is shown in Table 3.4, where SCISSOR has the highest F-MEASURE. It would be interesting to investigate the linguistic differences between GEOQUERY and GEO250 which may cause the performance difference, since GEOQUERY was collected from more diverse resources than GEO250 (Section 2.3).

To summarize, SCISSOR learns accurate semantic interpretation by utilizing the SAPT annotation. The main improvement of SCISSOR over other systems is on long sentences, where the annotated SAPTs provide the knowledge of accurate meaning composition structure. In Section 5.7.2, we shall give more detailed discussion on the strengths and weaknesses of utilizing the knowledge of syntax.

3.7 Related work

In syntactic parsing, Shen and Joshi (2005) presents an integrated statistical parser for a variant of LTAG, with a formalism of stronger generative capacity as compared to CFG. In semantic parsing, Zettlemoyer and Collins (2005, 2007) utilize the integrated treatment of syntax and semantics in CCG, and learn a semantic parser by learning a probabilistic CCG. In semantic role labeling, Yi and Palmer (2005) and Yi (2007) augment nodes in a syntactic parse with semantic argument information from PropBank (Palmer et al., 2005), and train two syntactic parers, Collins (1997)'s parser and Ratnaparkhi (1999)'s parser, directly on the augmented corpus. No modification is made to the two parsers: a syntactic and semantic label pair on SAPTs is treated as a single label in both of them[4]. Merlo and Musillo (2008) also explores an integrated approach in semantic role labeling, and achieves competitive results with other systems. Most recently, in information extraction, Finkel and Manning (2009) introduce a joint parser of information extraction and syntactic parsing, utilizing a discriminative CRF parser.

Semantic role labeling (SRL) (Gildea and Palmer, 2002) provides a layer of semantic information for semantic parsing, which can be used for improved generalization for semantic parsing (see Section 6.3).

3.8 Chapter Summary

SCISSOR learns statistical parsers that integrate syntax and semantics in order to produce a semantically augmented parse tree that is then used to compositionally generate a formal meaning representation. Experimental results in two

[4]There is small modification on Collins' parser for correctly finding syntactic heads using combined labels.

domains, a natural-language database interface and an interpreter for coaching instructions in robotic soccer, have demonstrated that SCISSOR generally produces accurate semantic representations. By augmenting a state-of-the-art statistical parsing model to include semantic information, it is able to integrate syntactic and semantic clues to produce a robust interpretation that supports the generation of complete formal meaning representations.

Chapter 4

Discriminative Reranking for Semantic Parsing

The generative model used in SCISSOR has a limited choice of features due to the nature of a generative model. Its performance can be potentially improved by using discriminative reranking, which explores arbitrary global features (Collins, 2000). In this chapter, we investigate discriminative reranking upon a baseline semantic parser, SCISSOR, where the composition of meaning representations is guided by syntax. We examine if global features used for reranking syntactic parsing can be adapted for semantic parsing by creating similar semantic features based on the mapping between syntax and semantics. We report experimental results on two real applications: an interpreter for coaching instructions in robotic soccer, and a natural-language database interface. The results show that reranking can improve the performance on the coaching interpreter, but not on the database interface where sentences are short, which are less likely for global features to show improvement on.

4.1 Motivation

The generative model in SCISSOR assumes that a SAPT is generated using a sequence of generation steps, and the probability of a SAPT is a product of the probabilities associated with these steps. Thus, it is often hard to incorporate discriminative features, since the choice of features is directly constrained by the choice of generation steps.

The performance of SCISSOR can be potentially improved by using discriminative reranking, which is distribution free and can explore arbitrary global features for reranking outputs from a baseline model. While reranking has benefited many tagging and parsing tasks (Collins, 2000, 2002c; Charniak and Johnson, 2005), including semantic role labeling (Toutanova et al., 2005), it has not yet been applied to semantic parsing before this work. In this chapter, we investigate the effect of discriminative reranking to semantic parsing.

We examine if the features used in reranking syntactic parses can be adapted for semantic parsing, more concretely, for reranking the top SAPTs from the baseline model SCISSOR. The syntactic features introduced by Collins (2000) for syntactic parsing are extended with similar semantic features, based on the coupling of syntax and semantics. The averaged perceptron (Collins, 2002a) is used as the reranking algorithm for this work, since it has been successfully applied to several tagging and parsing reranking tasks (Collins, 2002c,a). We present experimental results on two corpora: an interpreter for coaching instructions in robotic soccer (CLANG) and a natural-language database interface (Geoquery). The best reranking model significantly improves F-measure on CLANG from 82.3% to 85.1% (15.8% relative error reduction), however, it fails to show improvements on GEOQUERY where sentences are short, which are less likely for global features to show improvement on.

The organization of this chapter is as follows. Section 4.2 first describes the method for generating the n-best SAPTs with SCISSOR. Section 4.3 then introduces discriminate features for reranking SAPTs. After that, Section 4.4 gives experimental results.

4.2 Generating the n-best SAPTs with SCISSOR

We utilized the technique provided in Bikel's implementation of Collins' parser to generate the n-best SAPTs, which adopted the methodology in Collins (2000). In Collins' parsing models, dynamic programming is used in chart parsing to control the search space: when multiple derivations of a constituent result in the same history (conditioned context) for future generation steps, only the derivation with the highest probability is kept. Therefore, it would fail to generate the n-best parses. To achieve this, Collins (2000) turns off dynamic programming, and uses beam search instead, where a parse with its probability within a prune factor of the top probability in the same chart entry is kept.

In SCISSOR, the generation of semantic labels on modifiers is constrained by semantic subcategorization frames, for which data can be very sparse. Although this constraint improves SCISSOR's precision (which is important for semantic parsing), it also limits its recall. To generate plenty of candidate SAPTs for reranking, we extended the back-off levels for the parameters generating semantic labels of modifiers. The new set is shown in Table 4.1 using the parameters for the generation of the left-side modifiers as an example. The back-off levels 4 and 5 are newly added by removing the constraints from the semantic subcategorization. Although the best SAPTs found by the model may not be as precise as before, we expect that reranking can improve the results and rank correct SAPTs higher.

4.3 Features for Reranking SAPTs

By applying a reranking model on SCISSOR, we can make use of informative features not available in SCISSOR to discriminate between SAPTs that can lead to correct MRs and those that cannot. Intuitively, both syntactic and seman-

BACK-OFF LEVEL	$\Pr_{l_1}(L_i\|...)$
1	P,H,w,t,Δ,LC
2	P,H,t,Δ,LC
3	P,H,Δ,LC
4	P,H
5	P

Table 4.1: Extended back-off levels for the semantic parameter $\Pr_{l_1}(L_i|...)$ in Table 3.2.

tic features describing the syntactic and semantic sub-structures of a SAPT would be good indicators. Since the syntactic features introduced by Collins (2000) for reranking syntactic parse trees have been successful proven in both English and Spanish (Cowan and Collins, 2005), we examine if these syntactic features can be adapted for semantic parsing by creating similar syntactic and semantic features for reranking SAPTs. Besides the structural features, the log probability from the baseline model SCISSOR is also included as a feature, as in Collins (2000). A SAPT in CLANG is shown in Figure 4.1 for illustrating the features throughout this section.

4.3.1 Syntactic Features

All syntactic features introduced by Collins (2000) are included for reranking SAPTs. The full feature set is described in Section 2.2.1. For the convenience of introducing the corresponding semantic features later, we briefly describe several syntactic feature types:

1. Rules. These are the counts of unique syntactic context-free rules in a SAPT. The example in Figure 4.1 has the feature f(PRN $\rightarrow$ -LRB- NP COMMA NP -RRB-) = 1.

2. Bigrams. These are the counts of unique bigrams of syntactic labels in a

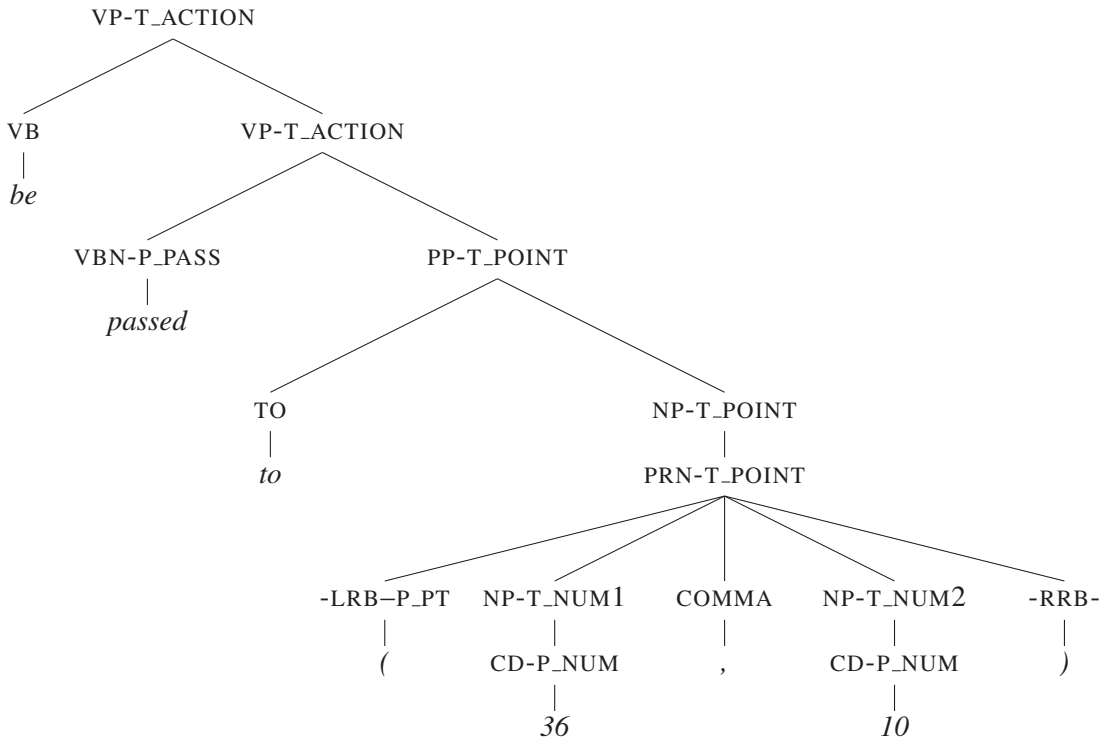


Figure 4.1: A SAPT for illustrating the reranking features. The comma's syntactic label "," is replaced by COMMA for a clearer description of features, and the NULL semantic labels are not shown. The syntactic and semantic heads of the rule expanding PRN-T_POINT are -LRB- and P_POINT.

constituent. They are also featured with the syntactic label of the constituent, and the bigram's relative direction (*left*, *right*) to the head of the constituent. The example in Figure 4.1 has the feature f(NP COMMA, *right*, PRN) = 1.

3. Grandparent Rules. These are the same as Rules, but also include the syntactic label above a rule. The example in Figure 4.1 has the feature f([PRN → -LRB- NP COMMA NP -RRB-], NP) = 1, where NP is the syntactic label above the rule "PRN → -LRB- NP COMMA NP -RRB-".

4. Grandparent Bigrams. These are the same as Bigrams, but also include the syntactic label above the constituent containing a bigram. The example in Figure 4.1 has the feature f([NP COMMA, *right*, PRN], NP) = 1, where NP is the syntactic label above the constituent PRN.

4.3.2 Semantic Features

Similarly, semantic features covering broader context of a SAPT are introduced for indicating a SAPT's semantic correctness. However, the tree structure in a SAPT can sometimes be more elaborate than needed for meaning composition, as the main argument for proposing semantic grammars (see Section 2.4.3). Therefore, we additionally introduce a set of semantic features which are extracted from a *pruned* SAPT with purely syntactic nodes removed.

4.3.2.1 Semantic Features from SAPTs

For a SAPT, a similar semantic feature type is introduced for each syntactic feature type in the syntactic feature set by replacing syntactic labels with semantic ones, where the semantic label NULL containing no meaning is not included. The corresponding semantic feature types for the features in Section 4.3.1 are:

1. Rules. The example in Figure 4.1 has the feature f(T_POINT $\rightarrow$ P_PT T_NUM1 T_NUM2) = 1.

2. Bigrams. The example has the feature f(T_NUM1 T_NUM2, *right*, P_PT) = 1, where the bigram {T_NUM1 T_NUM2} appears to the right of the semantic head P_PT.

3. Grandparent Rules. The example has the feature f([T_POINT $\rightarrow$ P_PT T_NUM1 T_NUM2], T_POINT) = 1, where the last T_POINT is the semantic label above the semantic rule T_POINT $\rightarrow$ P_PT T_NUM1 T_NUM2.

4. Grandparent Bigrams. The example has the feature f([T_NUM1 T_NUM2, *right*, T_POINT], T_POINT) = 1, where the last T_POINT is the semantic label above the T_POINT associated with the semantic label PRN.

We have also informally experimented with smoothed semantic features utilizing the domain ontology given by CLANG, which did not show improvements over reranking models not using these features.

4.3.2.2 Semantic Features from pruned SAPTs

Purely-syntactic structures in SAPTs exist with no meaning composition involved, such as the expansions from the syntactic label NP to PRN, and from PP to "TO NP" in Figure 4.1. Hence, one possible drawback of the semantic features derived directly from SAPTs is that they could include features with no meaning composition involved, which are intuitively not very useful. For example, the nodes with purely-syntactic expansions mentioned above would trigger a semantic rule feature with meaning unchanged (T_POINT $\rightarrow$ T_POINT). Another possible drawback of these features is that the features covering broader context could potentially

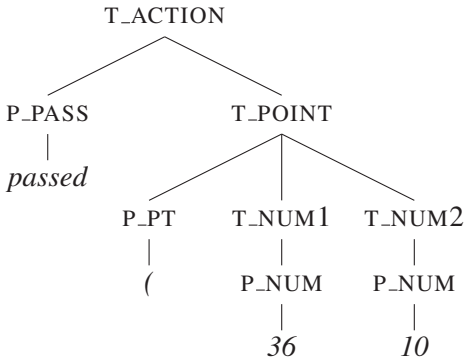


Figure 4.2: A pruned SAPT generated by removing purely-syntactic nodes from the SAPT in Figure 4.1 (with syntactic labels omitted.)

fail to capture the real high-level meaning composition information. For example, the Grandparent Rule example in Section 4.3.2.1 has T_POINT as the semantic grandparent of a P_PT composition, but not the real one T_ACTION.

To address these problems, another semantic feature set is introduced by deriving semantic features from trees where purely-syntactic nodes of SAPTs are removed (the resulting tree for the SAPT in Figure 4.1 is shown in Figure 4.2). In this tree representation, the example in Figure 4.2 would have the Grandparent Rule feature f([T_POINT$\rightarrow$ P_PT T_NUM1 T_NUM2], T_ACTION) = 1, with the correct semantic grandparent T_ACTION included.

4.4 Experimental Evaluation

4.4.1 Methodology

To test the reranking approach on SCISSOR, we experimented with it on both CLANG and GEO250 (Section 3.6.1). The detailed information of the corpora were shown in Section 2.3.

We adopted a standard 10-fold cross validation for evaluation: $\frac{9}{10}$ of the whole dataset was used for training (training set), and $\frac{1}{10}$ for testing (test set). To train a reranking model on a training set, a separate "internal" 10-fold cross validation over the training set was employed to generate n-best SAPTs for each training example using a baseline learner, where each training set was again separated into 10 folds with $\frac{9}{10}$ for training the baseline learner, and $\frac{1}{10}$ for producing the n-best SAPTs for training the reranker. Reranking models trained in this way ensure that the n-best SAPTs for each training example are not generated by a baseline model that has already seen that example and thus works well on that example. To test a reranking model on a test set, a baseline model trained on a whole training set was used to generate n-best SAPTs for each test example, and then the reranking model trained with the above method was used to choose a best SAPT from the candidate SAPTs. The performance of the parser was then measured in terms of *precision*, *recall*, and *F-measure* as in Section 3.6, and no partial credit was given for examples with partially-correct SAPTs.

When generating the n-best ($n = 50$) SAPTs, SCISSOR used a larger beam width than that used in Collins (2000) (10^3), with 10^8 for CLANG, and 10^{12} for GEO250. The reason to use a larger beam width for GEO250 is that the sentences in GEO250 are relatively short (6.87 words on average), thus it is harder to get enough candidates using a small beam width.

The averaged perceptron (see Section 2.2) was employed for training reranking models. To choose the correct SAPT of a training example required for training the averaged perceptron, we selected a SAPT that results in the correct MR; if multiple such SAPTs exist, the one with the highest baseline score was chosen. Since no partial credit was awarded in evaluation, a training example was discarded if it had no correct SAPT. Rerankers were trained on the 50-best SAPTs provided by

	CLANG			GEO250		
	Precision	Recall	F-measure	Precision	Recall	F-measure
SCISSOR	**89.5**	73.7	80.8	**98.5**	74.4	84.8
SCISSOR+	87.0	**78.0**	**82.3**	95.5	**77.2**	**85.4**

Table 4.2: The performance of the baseline model SCISSOR+ on CLANG and GEO250 compared with SCISSOR.

n	1	2	5	10	20	50
CLANG	78.0	81.3	83.0	84.0	85.0	85.3
GEO250	77.2	77.6	80.0	81.2	81.6	81.6

Table 4.3: Oracle recalls on CLANG and GEO250 as a function of number n of n-best SAPTs.

SCISSOR, and the number of perceptron iterations over the training examples was limited to 10. Typically, in order to avoid over-fitting, reranking features are filtered by removing the features which occur in very few training examples. We only removed features that never occurred in the training data since experiments with higher cut-offs failed to show any improvements.

4.4.2 Results

In this section, we describe the experiments with reranking models utilizing different feature sets, where all models include a SAPT's log probability assigned by the baseline model as a special feature.

First, the performance of the baseline learner SCISSOR was measured. Table 4.2 shows the results of SCISSOR using both the back-off levels in Figure 3.2 (SCISSOR) and the revised back-off levels in Section 4.3 (SCISSOR+). As expected, SCISSOR+ has better recall and worse precision than SCISSOR on both corpora due

	CLANG			GEO250		
	Precision	Recall	F-measure	Precision	Recall	F-measure
SCISSOR+	87.0	78.0	82.3	**95.5**	**77.2**	**85.4**
SYN	87.7	78.7	83.0	**95.5**	**77.2**	**85.4**
SEM_1	**90.0(23.1)**	**80.7(12.3)**	**85.1(15.8)**	95.5	76.8	85.1
$SYN+SEM_1$	89.6	80.3	84.7	95.5	76.4	84.9

Table 4.4: Reranking results on CLANG and GEO250 using different feature sets derived from SAPTs (with relative error reduction in parentheses).

to the additional levels of back-off. For all reranking experiments, SCISSOR+ is used as the baseline model.

Next, the optimal recalls a reranking model can possibly achieve were measured. Table 4.3 gives oracle recalls for CLANG and GEO250 where an oracle picks the correct parse from the n-best SAPTs if *any* of them are correct. Results are shown for increasing values of n. The optimal recall for a reranking model when n equals to 50 is 85.3% for CLANG, and 81.6% for GEO250.

Last, reranking models utilizing different feature sets were evaluated. Table 4.4 shows reranking results using different feature sets derived directly from SAPTs, where the reranking model SYN uses the syntactic feature set in Section 4.3.1, SEM_1 uses the semantic feature set in Section 4.3.2.1, and $SYN+SEM_1$ uses both. One observation is that in general, reranking improves the performance of semantic parsing on CLANG, but not on GEO250. This could be explained by the different oracle recall trends of CLANG and GEO250. We can see that in Table 4.3, even a small n can increase the oracle score on CLANG significantly, but not on GEO250. With the baseline score included as a feature, correct SAPTs closer to the top are more likely to be reranked to the top than the ones below, thus CLANG is more likely to have more sentences reranked correctly than GEO250. Another

	CLANG			GEO250		
	Precision	Recall	F-measure	Precision	Recall	F-measure
SEM_1	**90.0**	**80.7**	**85.1**	95.5	76.8	85.1
SEM_2	88.1	79.0	83.3	**96.0**	**77.2**	**85.6**
SEM_1+SEM_2	88.5	79.3	83.7	95.5	76.4	84.9
SYN+SEM_1	89.6	80.3	84.7	95.5	76.4	84.9
SYN+SEM_2	88.1	79.0	83.3	95.5	76.8	85.1
SYN+SEM_1+SEM_2	88.9	79.7	84.0	95.5	76.4	84.9

Table 4.5: Reranking results on CLANG and GEO250 comparing semantic features derived from both SAPTs and pruned SAPTs.

explanation is that global features utilized in reranking are not as effective on short sentences (6.87 words on average in GEO250) as on long sentences .

Another observation is that on CLANG, using semantic features greatly improves the performance of semantic parsing, as illustrated in both SEM_1 and SYN+SEM_1. Using SEM_1 alone achieves the best improvements over the baseline with 2.8% absolute improvement in F-measure (15.8% relative error reduction), which is significant at the 95% confidence level using a paired Student's t-test; using SYN+SEM_1 achieves similar performance: the difference between SEM_1 and SYN+SEM_1 is only one example. However, using syntactic features alone only slightly improves the results, because syntactic features do not directly discriminate between correct and incorrect meaning representations. To put this in perspective, Charniak and Johnson (2005) reported that reranking improves the F-measure of syntactic parsing from 89.7% to 91.0% with a 50-best oracle F-measure score of 96.8%.

Table 4.5 compares reranking results using semantic features derived from both SAPTs (SEM_1) and pruned SAPTs (SEM_2). It compares reranking models using these feature sets alone and together, and using them along with the syntactic

feature set (SYN) alone and together. Overall, SEM_1 provides better results than SEM_2 on CLANG and slightly worse results on GEO250 (only in one sentence), regardless of whether or not syntactic features are included. Using both semantic feature sets does not improve the results over just using SEM_1. On one hand, the better performance of SEM_1 on CLANG contradicts our expectations discussed in Section 4.3.2.2; the reason behind this needs to be investigated. On the other hand, it also suggests that the semantic features derived directly from SAPTs can provide good evidence for semantic correctness, even with the redundant purely-syntactically motivated features.

4.5 Related work

In semantic parsing, Lu et al. (2008) learns a probabilistic semantic grammar with a reranking model utilizing features similar to those in Collins (2000).

Discriminative reranking on semantic parsing can possibly be improved by using the progress on discriminative reranking. One direction of research in this area is to develop rich discriminative features. For example, in syntactic parsing, Charniak and Johnson (2005) propose features which describe parse sub-structures such as conjunction, constituent length and position. Collins (2002b) proposes kernel methods which can be applied efficiently to effectively utilize an exponential number of sub-structures in syntactic parses. We also plan to explore the features used in semantic role labeling (Gildea and Jurafsky, 2002; Carreras and Marquez, 2005), like the path between a target predicate and its argument.

There has also been an increased interest in finding better candidate outputs for reranking. One effort is on generating a better n-best list (Charniak and Johnson, 2005; Huang and Chiang, 2005). As pointed out by Charniak and Johnson (2005),

the beam search method used in Collins (2000) to find the n-best parses can prune away many good parses. Thus, Charniak and Johnson (2005) proposes a coarse-to-fine approach where dynamic programming for the n-best parses can be done feasibly on a set of pruned high-quality parse edges. Another effort is on reranking directly on a packed forest which compactly contains an exponential number of implicit parses (Huang, 2008), where non-local features are still made possible in dynamic programming by using an on-the-fly technique.

4.6 Chapter Summary

We have applied discriminative reranking to semantic parsing, where reranking features are developed from features for reranking syntactic parses based on the coupling of syntax and semantics. While the best reranking model significantly improves F-measure on a Robocup coaching task (CLANG) from 82.3% to 85.1%, it fails to improve the performance on a geography database query task (GEOQUERY), where sentences are short, and global features are less likely to show improvement.

Chapter 5

Semantic Parsing Using an Existing Syntactic Parser

This chapter describes a new approach to learning a semantic parser, SYN SEM, which exploits an existing syntactic parser to produce disambiguated parse trees that drive the compositional semantic interpretation. It also handles MRLs with logical variables. The resulting system produces accurate semantic interpretations on standard corpora on natural language interfaces for database querying and simulated robot control.

5.1 Motivation

Although SCISSOR (Chapter 3) learns to produce accurate semantic interpretations, it uses the extra annotation of SAPTs not required by other systems (Wong and Mooney, 2006; Kate and Mooney, 2006). In this chapter, we introduce a novel semantic parser, SYNSEM, which exploits an existing syntactic parser to produce disambiguated parse trees that drive the compositional semantic interpretation. With the advancement of statistical syntactic parsing, accurate syntactic parsers are available for many languages and could potentially be used to learn more effective semantic analyzers. Thus, this approach allows semantic parsing to conveniently leverage the progress in syntactic parsing.

Another improvement of SYNSEM over SCISSOR is that SYNSEM is capable of handling MRLs with logical variables. This improvement is necessary for two

reasons. First, due to the ability to deal with a wide range of linguistic phenomena such as quantification and modality, predicate logic languages, especially variants of first-order logic, have been fundamental MRLs for computational semantics for a long time (Montague, 1970; Thomason, 1974; Blackburn and Bos, 2005), and have been used extensively in semantic parsing (Zelle and Mooney, 1996; Copestake and Flickinger, 2000; Bos, 2005; Zettlemoyer and Collins, 2005, 2007; Wong and Mooney, 2007). Second, due to the availability of high-performance first-order theorem provers, using variants of first-order logic as MRLs would make possible automatic reasoning for natural language understanding, a key application of semantic parsing (Bos, 2006).

In addition, in SCISSOR, the meaning composition algorithm which transforms SAPTs to MRs (COMPOSEMR) is heuristic and deterministic, which may fail to explore alternative correct MRs. In response to this, in SYNSEM, semantic labels on SAPTs become more specific, where a predicate's missing arguments are encoded, and meaning composition is acurately specified in composition rules.

Last, since the accuracy of syntactic parsers is critical to SYNSEM, we also present the experiments for increasing robustness to syntactic parsing errors.

Specifically, SYNSEM uses standard compositional semantics to construct alternative MRs for a sentence based on its syntax tree, and then chooses the best MR based on a trained statistical disambiguation model. The learning system first employs a word alignment method from statistical machine translation (GIZA++ (Och and Ney, 2003)) to acquire a semantic lexicon that maps words to logical predicates. Then it induces rules for composing MRs and estimates the parameters of a maximum-entropy model for disambiguating semantic interpretations. We present experimental results on standard corpora demonstrating improved results on learning NL interfaces for database querying (GEOQUERY) and simulated robot

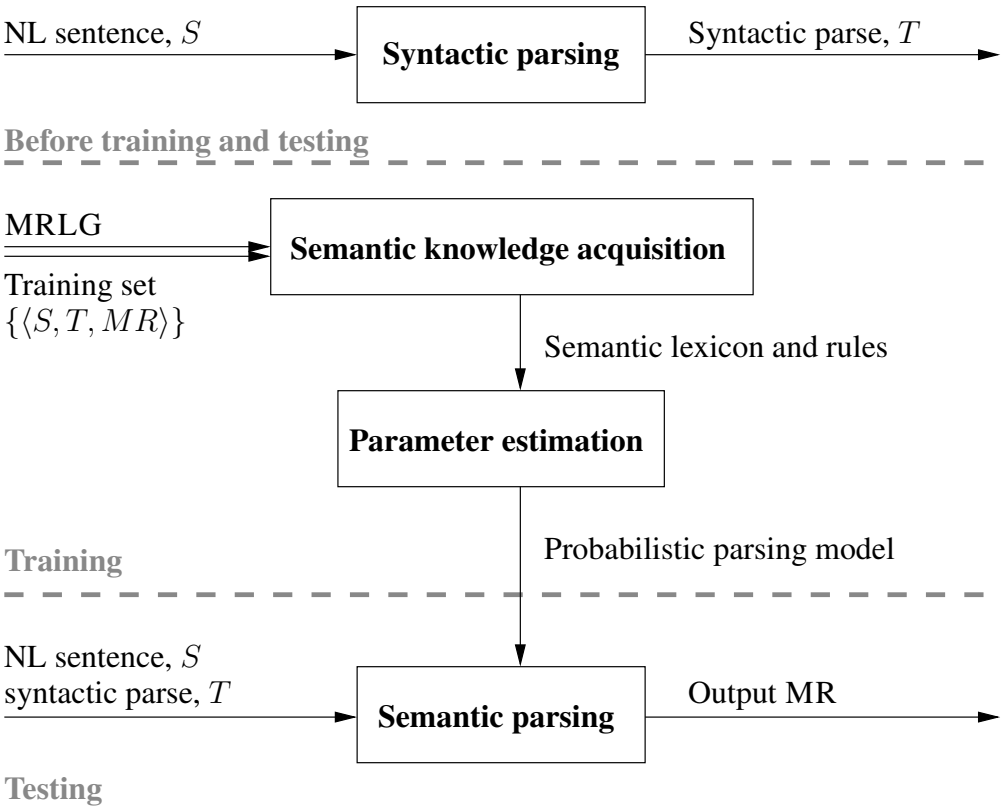


Figure 5.1: Overview of the SYNSEM semantic parsing algorithm

control (CLANG).

Figure 5.1 gives an overview of the SYNSEM semantic parsing algorithm. Before training, syntactic parses for all sentences in the training corpus are generated using an existing syntactic parser. During training, a semantic parser is trained on the set of NL-MR pairs together with the syntactic parses to learn a probabilistic semantic parsing model. During testing, test sentences together with their syntactic parses are parsed by the learned model to find the most likely MR.

Like SCISSOR, SYNSEM also assumes that an MRL is defined by an MRLG,

If our player 2 has the ball, then position our player 5 in the midfield.

```
((bowner (player our {2}))
 (do (player our {5}) (pos (midfield))))
```

Figure 5.2: A simple NL and its MR in the ROBOCUP domain.

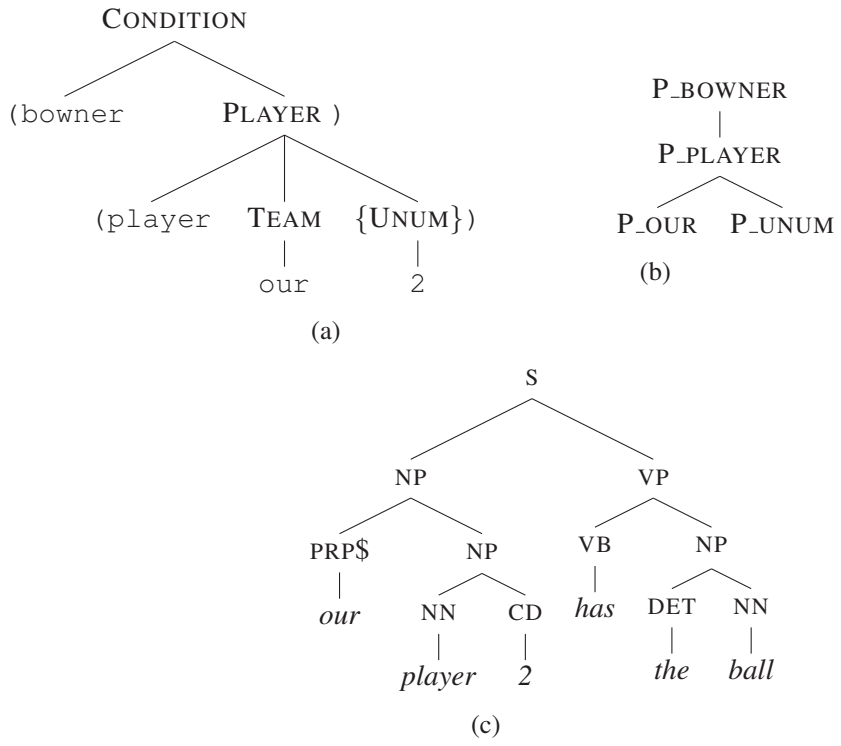


Figure 5.3: Parses for the condition part of the CLANG in Figure 5.2: (a) The parse of the MR (the nodes for parentheses are not separately shown for brevity). (b) The predicate argument structure of (a). (c) The parse of the NL.

PRODUCTION	PREDICATE
RULE→(CONDITION DIRECTIVE)	P_RULE
CONDITION→ (bowner PLAYER)	P_BOWNER
PLAYER→ (player TEAM {UNUM})	P_PLAYER
TEAM→our	P_OUR
UNUM→2	P_UNUM
DIRECTIVE→ (do PLAYER ACTION)	P_DO
ACTION→ (pos REGION)	P_POS
REGION→ (midfield)	P_MIDFIELD

Table 5.1: Sample production rules for parsing the CLANG example in Figure 5.2 and their corresponding predicates .

so that meaning representations can be uniquely parsed. In this chapter, the CLANG example used in Chapter 3 (Figure 3.1) shall be reused to demonstrate SYNSEM. For convenience, we include the example here in Figure 5.2. Figure 5.3 (a) and (b) show the condition part's MR parse and predicate-argument structure using the MRLG in Wong (2007). Sample MRLG productions and their predicates for parsing this example are shown in Table 5.1.

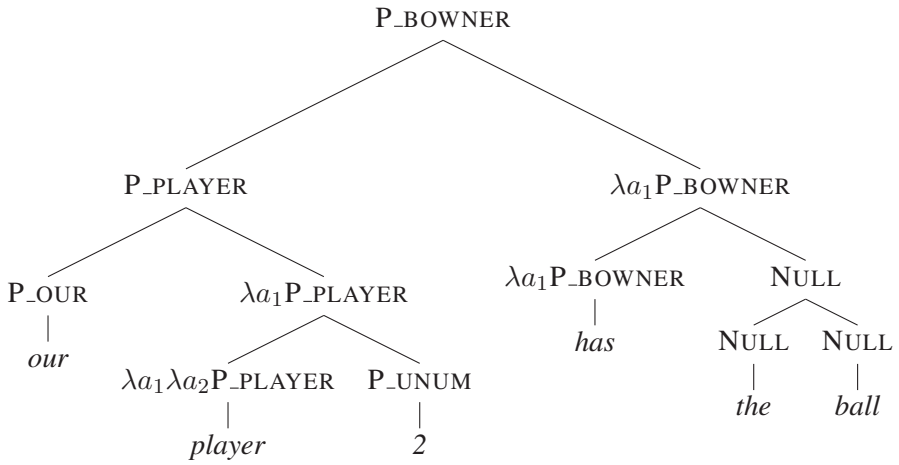
The remainder of the Chapter is organized as follows. Section 5.2 to Section 5.4 introduce the learning algorithm, assuming a variable-free MRL, including the basic framework (Section 5.2), the methodology for handling non-isomorphism between syntactic and MR parses (Section 5.3), and the process of learning semantic knowledge (Section 5.4). Based on this, Section 5.5 introduces the extension for handling logical forms. After that, Section 5.6 introduces the disambiguation model and Section 5.7 gives the experimental results. In addition, since the accuracy of syntactic parsers is critical to SYNSEM, Section 5.8 presents the experiments for increasing robustness to syntactic parsing errors.

5.2 Semantic Parsing Framework

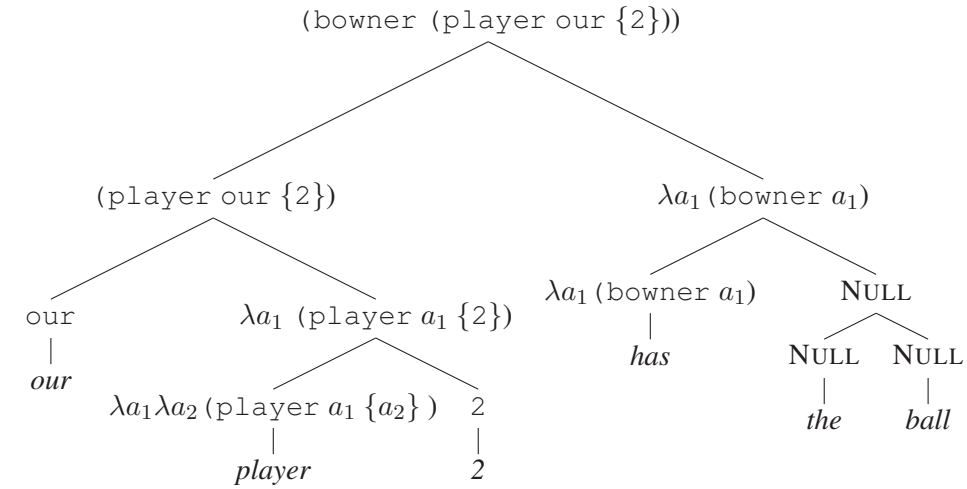
This section describes our basic framework, which is based on a fairly standard approach to computational semantics (Blackburn and Bos, 2005). The framework is composed of three components: 1) an existing syntactic parser to produce parse trees for NL sentences; 2) learned semantic knowledge (cf. Sec. 5.4), including a *semantic lexicon* to assign possible predicates (meanings) to words, and a set of *semantic composition rules* to construct possible MRs of a parent node on a syntactic parse given its children's MRs; and 3) a statistical disambiguation model (cf. Sec. 5.6) to choose among multiple possible semantic constructs as defined by the semantic knowledge.

The process of generating the semantic parse for an NL sentence is as follows. First, the syntactic parser produces a parse tree for the NL sentence. Second, the semantic lexicon assigns possible predicates for each word in the sentence. Third, all possible MRs of the sentence are constructed compositionally in a recursive, bottom-up fashion following its syntactic parse using composition rules. Lastly, the statistical disambiguation model scores each possible MR and returns the one with the highest score. Fig. 5.4 shows one possible SAPT for the condition part of the example in Fig. 5.2 given its syntactic parse in Fig. 5.3(c). A SAPT adds a semantic label to each non-leaf node in the syntactic parse tree. Unlike SCISSOR, besides the MRL predicate, the label also specifies the predicate's remaining (unfilled) arguments. The compositional process assumes a binary parse tree suitable for predicate-argument composition; parses in Penn-treebank style are binarized using Collins (1999) method.

Consider the construction of the SAPT in Fig. 5.4(a). First, each word is assigned a semantic label. Most words are assigned an MRL predicate. For example, the word *player* is assigned the predicate P_PLAYER with its two missing



(a) SAPT



(b) Semantic Derivation

Figure 5.4: Semantic parse for the condition part of the example in Fig. 5.2 using the syntactic parse in Fig. 5.3(c): (a) A SAPT with syntactic labels omitted for brevity. (b) The semantic derivation of the MR.

arguments, a_1 and a_2, indicated using λ. Words that do not introduce a predicate are given the label NULL, like *the* and *ball*.[1] Next, a semantic label is assigned to each internal node using learned composition rules that specify how arguments are filled when composing two MRs (cf. Sec. 5.4). The label λa_1P_PLAYER indicates that the remaining argument a_2 of the P_PLAYER child is filled by the MR of the other child (labeled P_UNUM).

Finally, the SAPT is used to guide the composition of the sentence's MR. At each internal node, an MR for the node is built from the MRs of its children by filling an argument of a predicate, as illustrated in the semantic derivation shown in Fig. 5.4(b). Semantic composition rules (cf. Sec. 5.4) are used to specify the argument to be filled. For the node spanning *player 2*, the predicate P_PLAYER and its second argument P_UNUM are composed to form the MR: λa_1 (`player` a_1 {`2`}). Composing an MR with NULL leaves the MR unchanged. An MR is said to be *complete* when it contains no remaining λ variables. This process continues up the tree until a complete MR for the entire sentence is constructed at the root.

5.3 Ensuring Meaning Composition

The basic compositional method in Sec. 5.2 only works if the syntactic parse tree strictly follows the predicate-argument structure of the MR, since meaning composition at each node is assumed to combine a predicate with one of its arguments. However, this assumption is often not satisfied for at least two reasons. First, predicates and arguments can be detached in various linguistic phenomena

[1]The words *the* and *ball* are not truly "meaningless" since the predicate P_BOWNER (ball owner) is conveyed by the phrase *has the ball*. For simplicity, predicates are introduced by a single word, but statistical disambiguation (cf. Sec. 5.6) uses surrounding words to choose a meaning for a word whose lexicon entry contains multiple possible predicates.

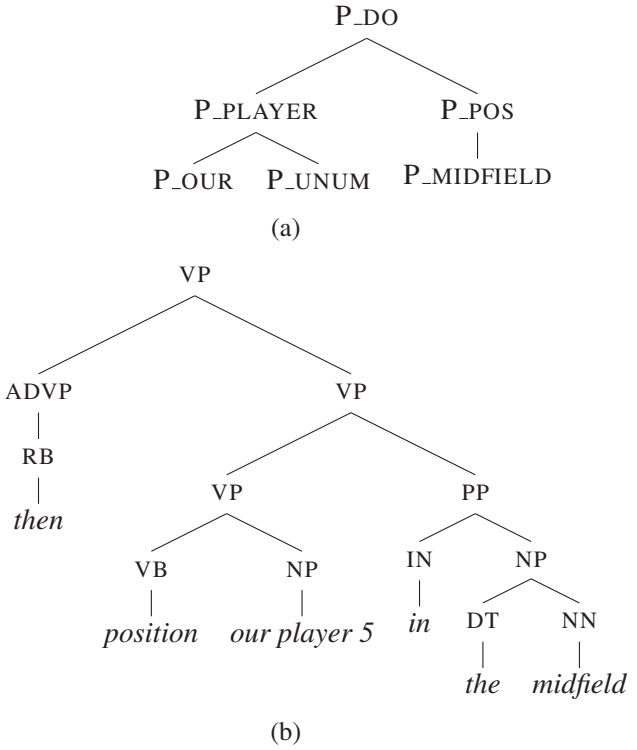


Figure 5.5: Parses for the directive part of the CLANG in Fig. 5.2 (the nodes for parentheses are not separately shown for brevity): (a) The predicate-argument structure of the MR. (b) The parse of the NL (the parse of the phrase *our player 5* is omitted for brevity).

such as a moved subject or object in questions and relative clauses (Steedman and Baldridge, to appear). Second, syntactic parses of an NL and MRL may be non-isomorphic when seen as two different languages describing the same language-independent world, as demonstrated in machine translation (Yamada and Knight, 2001). For example, the node covering *the ball* in the NL parse (Figure 5.3(c)), arguably, does not have a matching node in the MR parse (Figure 5.3(b)). Another example is shown in the non-isomorphic NL and MRL parses for the directive part of the example in Figure 5.2 (see Figure 5.5). In the MR parse, P_POS (*position*) is composed with P_PLAYER (*our player 5*) after it has its argument P_MIDFIELD (*midfield*) filled; whereas in the NL parse, it is composed with P_PLAYER before it has P_MIDFIELD filled. Besides, to be robust to syntactic errors, SYNSEM learns to construct correct MRs from syntactic parses with errors, which often cause this non-isomorphic phenomena.

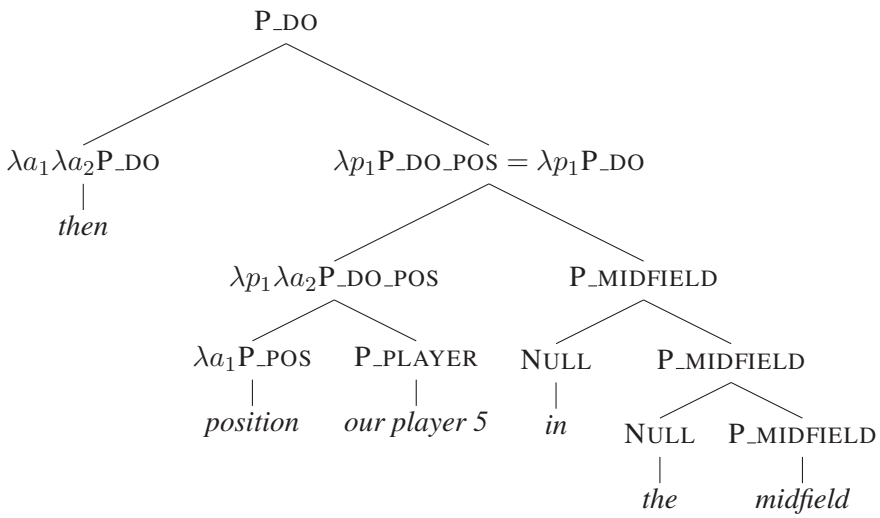
To ensure meaning composition in this case, we automatically create *macro-predicates* as meaning postulates, that combine multiple predicates into one, so that the child nodes' MRs can be composed as arguments or internal predicates to a macro-predicate. Fig. 5.7 shows the macro-predicate P_DO_POS (DIRECTIVE→`(do` PLAYER `(pos` REGION`))`) formed by merging the P_DO and P_POS in Fig. 5.5(a). The macro-predicate has two arguments, one of type PLAYER (a_1) and one of type REGION (a_2). Now, P_POS and P_PLAYER can be composed as arguments to this macro-predicate as shown in Fig. 5.6(b). However, it requires assuming a P_DO predicate that has not been formally introduced. To indicate this, a lambda variable, p_1, is introduced that ranges over predicates and is provisionally bound to P_DO, as indicated in Fig. 5.6(b) using the notation p_1:`do`. Eventually, this predicate variable must be bound to a matching predicate introduced from the lexicon. In the example, p_1:`do` is eventually bound to the P_DO predicate introduced by the word *then* to

form a complete MR.

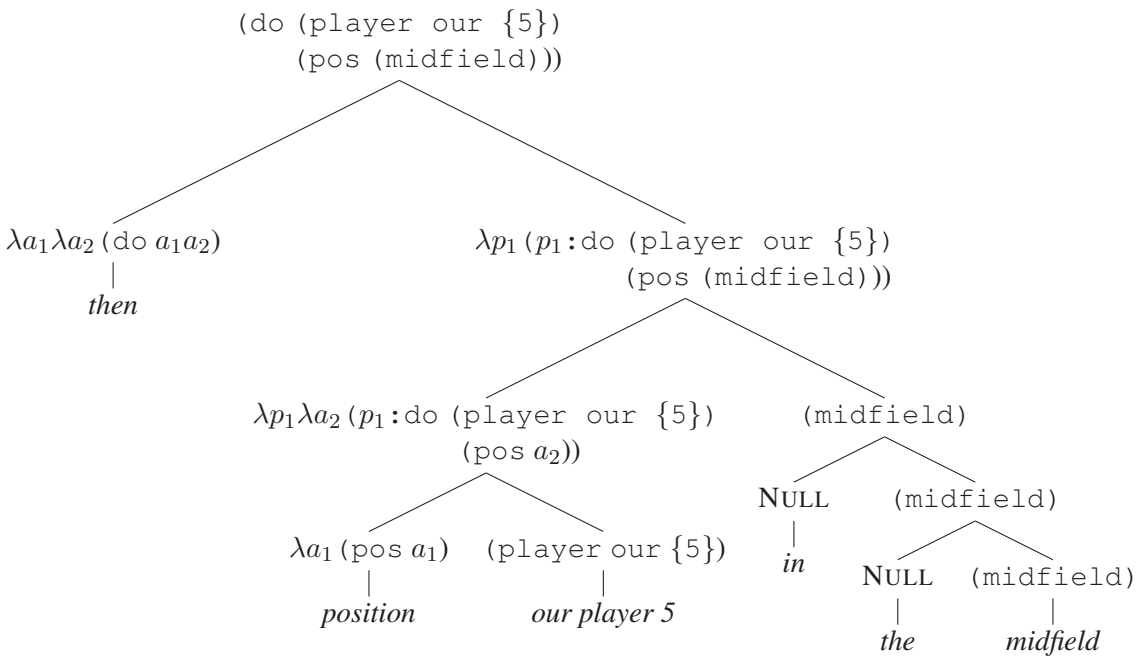
Macro-predicates are introduced as needed during training in order to ensure that each MR in the training set can be composed using the syntactic parse of its corresponding NL giving reasonable assignments of predicates to words. For each SAPT node that does not combine a predicate with a legal argument, a macro-predicate is formed by merging all predicates on the paths from the child predicates to their lowest common ancestor (LCA) in the MR parse. Specifically, a child MR becomes an argument of the macro-predicate if it is complete (i.e. contains no λ variables); otherwise, it also becomes part of the macro-predicate and its λ variables become additional arguments of the macro-predicate. For the node spanning *position our player 5* in the example, the LCA of the children P_PLAYER and P_POS is their immediate parent P_DO, therefore P_DO is included in the macro-predicate. The complete child P_PLAYER becomes the first argument of the macro-predicate. The incomplete child P_POS is added to the macro-predicate P_DO_POS and its λ variable becomes another argument.

For improved generalization, once a predicate in a macro-predicate becomes complete, it is removed from the corresponding macro-predicate label in the SAPT. For the node spanning *position our player 5 in the midfield* in Fig. 5.6(a), P_DO_POS becomes P_DO once the arguments of `pos` are filled.

Previously, a number of works in machine translation and semantic parsing have also addressed the non-isomorphism problem. In machine translation, Yamada and Knight (2001) flatten subtrees for maintaining parse tree isomorphism in syntax-based machine translation; Shieber and Schabes (1990) and Eisner (2003) use synchronous tree-substitution grammars and synchronous tree-adjoining grammars to maintain tree structure. In semantic parsing, Wong and Mooney (2006) use an approach similar to that of Yamada and Knight (2001) which merges nodes in an



(a) SAPT



(b) Semantic Derivation

Figure 5.6: Semantic parse for the directive part of the example in Fig. 5.2 using the syntactic parse in Fig. 5.5(b): (a) A SAPT with syntactic labels omitted for brevity. (b) The semantic derivation of the MR.

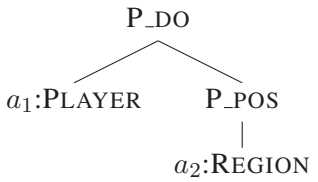


Figure 5.7: The predicate-argument structure of macro-predicate P_DO_POS.

MR parse tree to maintain parse tree isomorphism. Zettlemoyer and Collins (2007) introduce non-standard CCG combinators that relax the grammar to handle certain phenomena such as the deletion of a word. When applied to a new NL-MRL pair, new combinators may need to be introduced. To handle cases of non-isomorphism such as in Figure 5.6 where P_POS (syntactic head) takes P_PLAYER as an argument in syntax but not in the MR, larger lexical categories may need to be introduced. Our approach is driven by existing syntactic parses which ensures that the MRs of a node's children can be composed to form the MRs of the node according to the gold-standard MR. In the future, we plan to investigate the similarity between SYNSEM and construction grammar (Goldberg, 1995), which defines constructions as linguistic units that necessarily have some non-compositional semantics.

In the following two sections, we describe the two subtasks of inducing a semantic grammar and disambiguation model for learning the enhanced compositional framework. Both subtasks require a training set of NLs paired with their MRs. Each NL sentence also requires a syntactic parse generated using Bikel (2004)'s implementation of Collins parsing model 2.

5.4 Learning Semantic Knowledge

Learning semantic knowledge starts from learning the mapping from words to predicates. We use an approach based on Wong and Mooney (2006), which constructs word alignments between NL sentences and their MRs. Normally, word alignment is used in statistical machine translation to match words in one NL to words in another; here it is used to align words with predicates based on a "parallel corpus" of NL sentences and MRs. We assume that each word alignment defines a possible mapping from words to predicates for building a SAPT and semantic derivation which compose the correct MR. Semantic lexicon and composition rules are then extracted directly from each of the nodes of the resulting semantic derivations.

Generating word alignments for each training example proceeds as follows. First, each MR in the training corpus is parsed using the MRLG. Next, each resulting parse tree is linearized to produce a sequence of predicates by using a top-down, left-to-right traversal of the parse tree. Then the GIZA++ implementation (Och and Ney, 2003) of IBM Model 5 is used to generate the five best word/predicate alignments from the corpus of NL sentences each paired with the predicate sequence for its MR.

After predicates are assigned to words using word alignment, for each alignment of a training example and its syntactic parse, a SAPT is generated for composing the correct MR, using the processes discussed in Sections 5.2 and 5.3. Specifically, a semantic label is assigned to a parent node of a SAPT, so that the MRs of its children are composed correctly, according to the MR for this example.

There are two cases that require special handling. First, when a predicate is not aligned to any word, the predicate must be inferred from context. For example,

in CLANG, *our player* is frequently just referred to as *player* and the `our` must be inferred.

When building a SAPT for such an alignment, the meaning composition algorithm learns what arguments of a predicate can have default value, and uses them when needed. Second, when a predicate is aligned to several words, i.e. it is represented by a phrase, then the alignment is transformed into several alignments where each predicate is aligned to each single word in order to fit the assumptions of compositional semantics.

Given the SAPTs constructed from the results of word-alignment, a semantic derivation for each training sentence is constructed using the methods described in Sections 5.2 and 5.3. Composition rules are then extracted from these derivations.

Formally, composition rules are of the form:

$$\Lambda_1.P_1 + \Lambda_2.P_2 \Rightarrow \Lambda_p.P_p, R \tag{5.1}$$

where P_1, P_2 and P_p are predicates for the left child, right child, and parent node, respectively. Each predicate includes a lambda term Λ of the form $\{\lambda p_{i_1}, \ldots, \lambda p_{i_m}, \lambda a_{j_1}, \ldots, \lambda a_{j_n}\}$, a list of all missing predicate and argument variables for the predicate. The component R specifies how some arguments of the parent predicate are filled when composing the MR for the parent node. It is of the form: $\{a_{k_1}{=}R_1, \ldots, a_{k_l}{=}R_l\}$, where R_i can be either a child (c_i), or a child's complete argument (c_i, a_j) if the child itself is not complete.

For instance, the rule extracted for the node for *player 2* in Fig. 5.4(b) is:

$\lambda a_1 \lambda a_2$.P_PLAYER + P_UNUM $\Rightarrow \lambda a_1$.P_PLAYER, $a_2{=}c_2$,

and for *position our player 5* in Fig. 5.6(b):

λa_1.P_POS + P_PLAYER $\Rightarrow$ $\lambda p_1 \lambda a_2$.P_DO_POS, a_1=c_2,

and for *position our player 5 in the midfield*:

$\lambda p_1 \lambda a_2$.P_DO_POS + P_MIDFIELD $\Rightarrow$ λp_1.P_DO_POS, $\{a_1$=(c_1,a_1), a_2=$c_2\}$.

Learning semantic knowledge is necessary for handling ambiguity such as word sense and semantic roles. It is also used to ensure that an MR is a legal expression in the MRL.

5.5 Semantic Parsing with Logical Forms

The SYNSEM semantic parsing algorithm discussed so far assumes MRLs free of logical variables. In this section, we extend the algorithm to use predicate logic as MRLs where logical variables play a significant role.

But before turning into the details of the extension, let's first briefly review how meanings are represented in predicate logic informally[2]. In predicate logic, the most basic meaning elements are constants for basic objects (e.g. `texas`), logical variables (e.g. x_1, x_2 in Figure 5.8) and predicates (e.g. `answer`, `river` and `loc`), where logical variables denote a set of entities, and predicates denote relations and functions over their arguments in the application domain. These basic elements are then composed to form complex formulas either by forming predicate-argument relations or by connecting subformulas using logical connectives, where variable names can be shared among predicates to constrain the exact entities represented.

[2]Although we mainly focus on the logical query language in the GEOQUERY domain, the algorithm developed here is also applicable to other logical languages.

What are the rivers in Texas?
answer(x_1,(river(x_1),loc(x_1,x_2),equal(x_2,stateid(texas))))

Figure 5.8: A sample NL and its MR in the GEOQUERY domain.

Consider the sample query in Figure 5.8 in the GEOQUERY domain. The logical variable x_2 denotes a basic entity in the application domain, the state of Texas, and the first-order predicate `loc` denotes a binary relation asserting if the first entity is located in the second. The predicate `answer` is a higher-order predicate which takes the conjunction (,) of the predicates `river`, `loc` and `equal` as its only argument, specifying the conditions the target variable x_1 has to satisfy. Note that x_1 is shared among `river` and `loc` to form the exact subset of entities it represents: the rivers in Texas; it later also appears in ANSWER to specify the entities to return as an answer.

Hence, to use predicate logic as target MRL languages, our algorithm should have the capability to model predicate-argument relations, dependencies among logical variables, and logical connectives. The modeling of predicate-argument relations has already been implemented in our existing algorithm. In this section, we show how the remaining tasks can be further incorporated, by first defining the extended MRLG and semantic parsing framework for predicate logic, and then describing the process of learning the extended semantic knowledge.

5.5.1 Predicate Logic as Meaning Representation Language

In this section, we describe the extended MRLG for predicate logic, focusing on logical variables and connectives. The predicate logic language we considered is the logical query language used in GEOQUERY, which is based on Prolog consisting of both first-order and higher-order predicates (Zelle and Mooney, 1996).

In an extended MRLG, production rules are extended for logical variables.

PRODUCTION	PREDICATE
QUERY → answer(v_1,FORM)	P_ANSWER
FORM → river(v_1)	P_RIVER
FORM → loc(v_1,v_2)	P_LOC
FORM → equal(v_1,stateid(texas))	P_EQUAL
FORM → (FORM,FORM,FORM)	

Table 5.2: The production rules for parsing the GEOQUERY example in Figure 5.8 and their corresponding predicates.

How many major cities are in states bordering Texas?

```
answer(x1,count(x2,(city(x2),major(x2),loc(x2,x3),
                    state(x3),next_to(x3,x4),
                    equal(x4,stateid(texas)))))
```

Figure 5.9: The NL/MRL non-isomorphism example used in Wong and Mooney (2007).

The LHS of a production rule is still a nonterminal, however, its RHS can be strings of not only terminals and non-terminals, but also logical variables, where unique variables from left to right are annotated with $v_1, v_2, \ldots$ sequentially. When forming an MR, these variables are substituted for names (annotated with $x_1, x_2, \ldots$), which may be shared among predicates. Table 5.2 shows the production rules in the MRLG for parsing the example in Figure 5.8.

Production rules for parsing connectives are also added to an MRLG. In the GEOQUERY logical language, conjunction rules are added for the logical conjunction. The last row in Table 5.2 shows the rule for parsing the conjunction in Figure 5.8. When having multiple conjunction rules in an MRLG, it may fail to generate a unique parse for an MR, since a conjunction constituent with multiple children (> 2) can be further broken down into conjunctions with smaller arity. In this case, the grammar chooses the parse where all conjunctive parts are in the same constituent. Figure 5.10(a) shows such a parse for the MR in Figure 5.8 using the

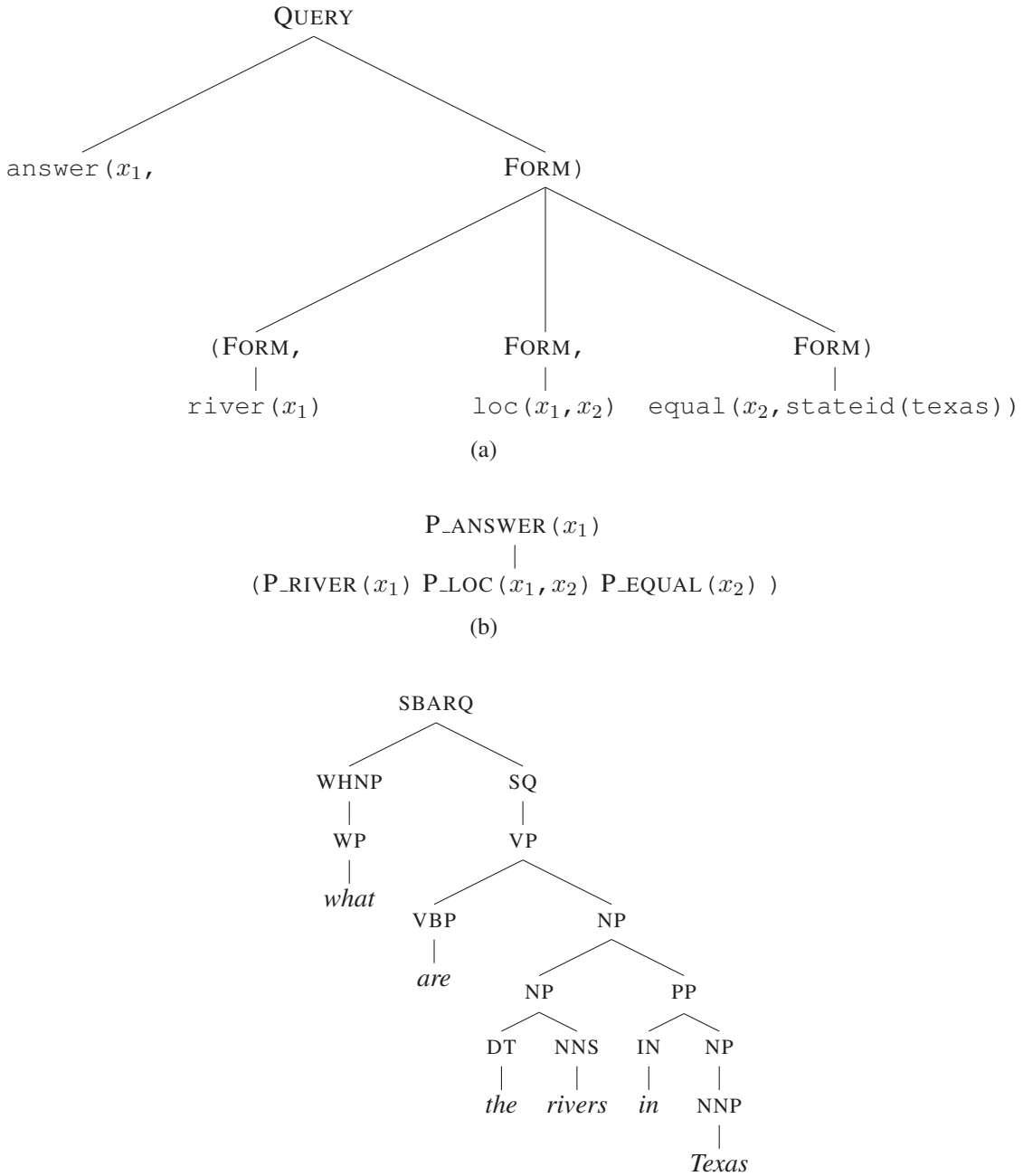


Figure 5.10: Parses for the GEOQUERY example in Figure 5.8: (a) The parse of the MR. (b) The predicate argument structure of (a). (c) The parse of the NL.

MRLG in Wong (2007), augmented with n-ary conjunction rules.

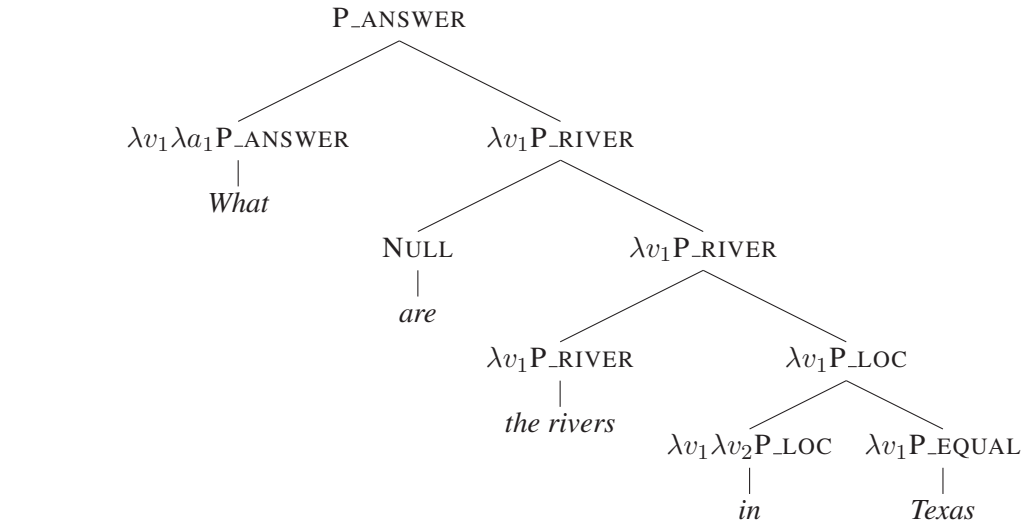
Unlike our approach, Wong (2007) only allows binary conjunction so that a n-ary conjunction in an MR is parsed into a binary tree where the internal structure of the conjunction is assumed. This may cause unnecessary parse tree non-isomorphism since it forces a constraint not present in an MR (see Figure 5.9 for an illustrative example in Wong and Mooney (2007)). In our approach, the grammar does not assume the order of conjunctive parts.

Figure 5.10(b) shows the corresponding predicate-argument structure of Figure 5.10(a). In this structure, the only argument of the predicate P_ANSWER is the conjunction of P_RIVER, P_LOC and P_EQUAL; each predicate is followed by a list of names substituted for its logical variables. Note the name x_1 is shared among the predicates P_ANSWER, P_RIVER and P_LOC representing the same entities.

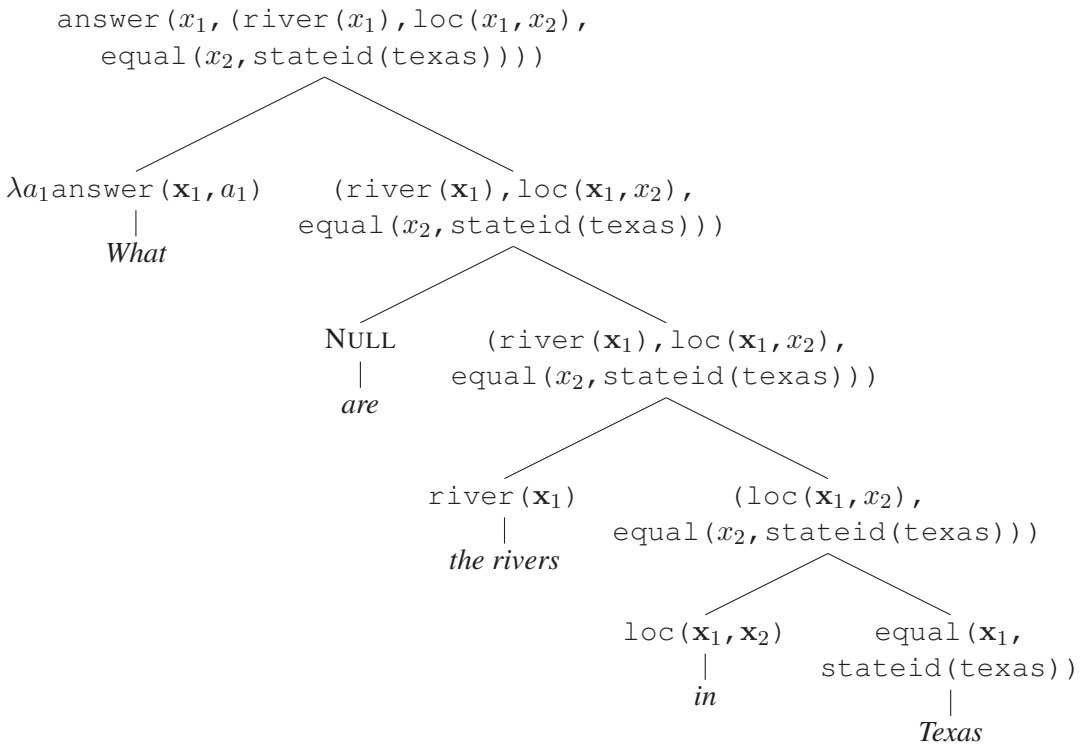
5.5.2 Semantic Parsing Framework

Our SYNSEM semantic parsing framework relies on a semantic lexicon to assign semantic labels to words, and a set of semantic composition rules to assign semantic labels to internal SAPT nodes and construct MRs. In this section, we show the extension to both components for handling logical forms.

We introduce the logical-variable binding operators λv to a semantic label, which bind occurrences of logical variables in a predicate. Similar to the existing λ operators for missing arguments and internal predicates, these operators specify the logical variables to be shared with other predicates (*shared logical variable*). Here are some sample lexicon entries for building the correct MR in example 5.8 (see the SAPT in 5.11(a)):



(a) SAPT



(b) Semantic Derivation

Figure 5.11: Semantic parse for the GEOQUERY example in Fig. 5.8 using the syntactic parse in Fig. 5.10(c).

$$
\begin{aligned}
\textit{What} &= \lambda v_1 \lambda a_1 \text{P\_ANSWER} \\
\textit{river} &= \lambda v_1 \text{P\_RIVER} \\
\textit{in} &= \lambda v_1 \lambda v_2 \text{P\_LOC} \\
\textit{Texas} &= \lambda v_1 \text{P\_EQUAL}
\end{aligned}
$$

where in the semantic label $\lambda v_1 \lambda v_2$P_LOC, $\lambda v_1 \lambda v_2$ bind the corresponding logical varibles in P_LOC (FORM $\rightarrow$ `loc(`v_1`,`v_2`)`). When building an MR, logical variables $v_1, v_2, \ldots$ in a predicate are initially named $x_1, x_2, \ldots$ sequentially.

Composition rules in Equation 5.1 are also extended accordingly:

$$\Lambda_1.P_1(N_1) + \Lambda_2.P_2(N_2) \Rightarrow \Lambda_p.P_p[N_p], R \tag{5.2}$$

where $\Lambda_1.P_1$, $\Lambda_2.P_2$, and $\Lambda_p.P_p$ are extended semantic labels for child and parent nodes, with the shared logical variables λv included. The term N_p is a list of names for all logical variables in P_p, including both λv variables and other unbound variables (free variables), where $\{v_1, v_2, \ldots\}$ are named $\{x_1, x_2, \ldots\}$ sequentially[3]. N_i ($i = 1, 2$) is a list of names for P_i's λv variables after applying the rule, which are their shared names in the parent predicate. This is made possible due to an assumption that we shall elaborate in Section 5.5.3: all child predicates' λv variables must be shared with their parent predicate, either as its λv variables or free variables. Before applying the rule, if some name in N_i ($i = 1, 2$) is already used in the MR of that child node, it should be renamed to avoid accidental binding of logical variables, known as α-*conversion* in lambda calculus (Blackburn and Bos, 2005).

Before showing sample rules, let's introduce one special type of composition rule for handling conjunction:

$$\Lambda_1.P_1(N_1) + \Lambda_2.P_2(N_2) \Rightarrow_c \Lambda_p.P_p[N_p] \tag{5.3}$$

[3]These names can actually be omitted, but are shown for clarity.

where the subscript $_c$ in $\Rightarrow_c$ stands for conjunction. Conjunction rules contain no component R for specifying how arguments of a parent predicate are filled when composing MR. Instead, the MR of a parent node is simply the conjunction of its children's MRs with name substitution applied.

Here are some sample composition rules for building the correct MR in example 5.8 (see the SAPT and semantic derivation in Figure 5.11):

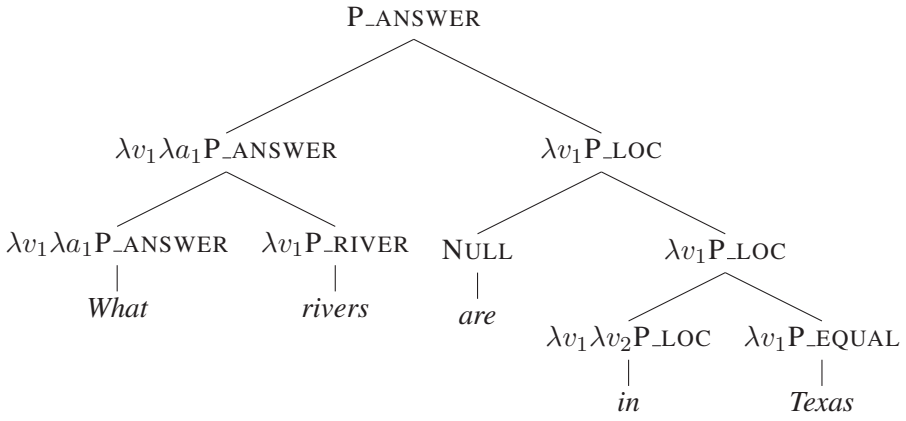
$$\lambda v_1 \lambda v_2 \text{P\_LOC}\ (x_1,x_2) + \lambda v_1 \text{P\_EQUAL}\ (x_2) \Rightarrow_c \lambda v_1 \text{P\_LOC}\ [x_1,x_2]$$
$$\lambda v_1 \text{P\_RIVER}\ (x_1) + \lambda v_1 \text{P\_LOC}\ (x_1) \Rightarrow_c \lambda v_1 \text{P\_RIVER}\ [x_1]$$
$$\lambda v_1 \lambda a_1 \text{P\_ANSWER}\ (x_1) + \lambda v_1 \text{P\_RIVER}\ (x_1) \Rightarrow \text{P\_ANSWER}\ [x_1], \{a_1 = c_2\}$$

In predicate logic, a higher-order predicate's argument can be a conjunction of multiple child predicates (see Figure 5.9). Often in an NL's syntax parse, such a parent predicate may be attached with one child predicate at a time, before they form a complete argument. To handle this situation, we allow a predicate's argument to be partially filled in composition rules. A missing argument variable is attached to a predicate until all of its child predicates are found. Figure 5.12 gives an example for generating the MR in Figure 5.8, where the following composition rules are used:

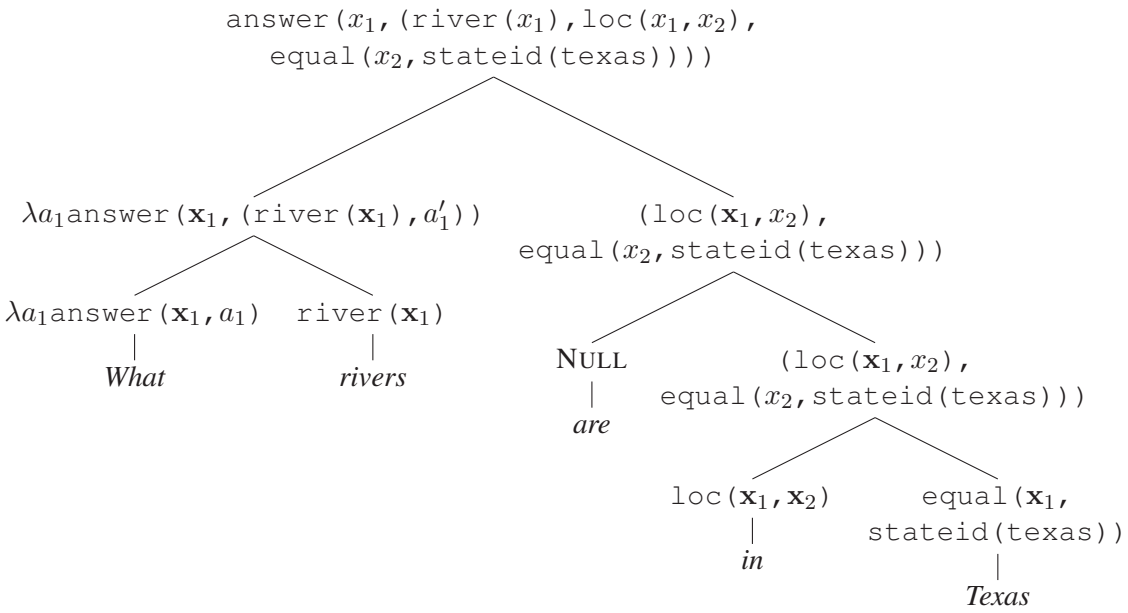
$$\lambda v_1 \lambda a_1 \text{P\_ANSWER}\ (x_1) + \lambda v_1 \text{P\_RIVER}\ (x_1) \Rightarrow \lambda v_1 \lambda a_1 \text{P\_ANSWER}\ [x_1], \{a_1 = c_2\}$$
$$\lambda v_1 \lambda a_1 \text{P\_ANSWER}\ (x_1) + \lambda v_1 \text{P\_LOC}\ (x_1) \Rightarrow \text{P\_ANSWER}\ [x_1], \{a_1 = c_2\}$$

5.5.3 Learning Semantic Knowledge

Learning semantic knowledge for logical forms uses the same process as described in Section 5.4. First, word alignments between words in an NL sentence and predicates in the linearized MR parse are constructed using a word alignment model. Next, a SAPT and semantic derivation which compose the correct MR are built from an NL's syntactic parse and a word alignment. Last, semantic lexicon



(a) SAPT



(b) Semantic Derivation

Figure 5.12: An example of an incrementally-filled argument for generating the MR in Fig. 5.8.

and composition rules are extracted directly from the nodes of the resulting SAPTs and semantic derivations.

Given a syntactic parse and word alignment, a key process for generating a correct SAPT is to attach to each node a correct semantic label, including both a predicate and its λ variables. To decide the predicate, a compositional variable-dependency assumption is required besides using the methodology in Section 5.4: a child predicate's λv variables must be shared with the parent predicate, either as its λv variables or free variables. Once a predicate is decided for a node, its λv variables are merely the predicate's logical variables whose names are not fully covered by the MR of the current node. We illustrate this process in the following two cases.

In conjunction, a parent predicate is chosen to be the child predicate whose names of λv variables consume the other child predicate's names. For example, the child predicate P_LOC in Figure 5.11(a) is the parent predicate for the constituent *in Texas*, since the names of its λv variables, $\{x_1, x_2\}$, consumes the other child predicate P_EQUAL's names, $\{x_2\}$. Its attached λv list is $\{\lambda v_1\}$ where v_1 is bound, since its name x_1 still appears in the predicates P_RIVER and P_ANSWER which are not covered by the current MR, while v_2 is not bound since all appearances of its name x_2 are completely covered. Sometimes, both child predicates may have the same name set used for their λv variables. Then, the one on the syntactic head becomes the parent predicate (e.g. the constituent *the rivers in Texas*).

In non-conjunction, a parent predicate is chosen using the methodology in Section 5.4. Occasionally, special handling is required to make sure that a child predicate's λv variables are shared with its parent predicate. Consider composing *smallest state* in the following example,

What is the smallest state by area?

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

where the word *smallest* stands for the parent predicate P_SMALLEST (in the MR, smallest(x_2, FORM)), and *state* stands for the child predicate P_STATE (state (x_1)). Although x_1 in P_STATE needs to be shared with other predicates in the MR, it is not a name for the parent predicate P_SMALLEST's logical variable. In this case, a logical variable invisible to the MR is introduced to the parent predicate to make the assumption true.

Once the correct SAPTs and semantic derivations are generated, semantic lexicon and composition rules are extracted straightforwardly from the nodes of the resulting SAPTs and semantic derivations. The following conjunction rule can be extracted from the node *in Texas*:

$$\lambda v_1 \lambda v_2 \text{P\_LOC}\ (x_1,x_2) + \lambda v_1 \text{P\_EQUAL}\ (x_2) \Rightarrow_c \lambda v_1 \text{P\_LOC}\ [x_1,x_2]$$

where the names attached to each child predicate are decided to form the correct MR (loc(x_1,x_2),equal(x_2,stateid(texas))).

5.6 Learning a Disambiguation Model

Usually, multiple possible semantic derivations for an NL sentence are warranted by the acquired semantic knowledge, thus disambiguation is needed. To learn a disambiguation model, the learned semantic knowledge (see Section 5.4) is applied to each training example to generate all possible semantic derivations for an NL sentence given its syntactic parse. Here, unique word alignments are not required, and compositional and possible non-compositional constructs compete for the best semantic parse.

We use a maximum-entropy model similar to that of Zettlemoyer and Collins (2005) and Wong and Mooney (2006). The model defines a conditional probability distribution over semantic derivations (D) given an NL sentence S and its syntactic parse T:

$$\Pr(D|S,T;\bar{\theta}) = \frac{\exp\sum_i \theta_i f_i(D)}{Z_{\bar{\theta}}(S,T)} \tag{5.4}$$

where $\bar{f}$ $(f_1,\ldots,f_n)$ is a feature vector parameterized by $\bar{\theta}$, and $Z_{\bar{\theta}}(S,T)$ is a normalizing factor. Three simple types of features are used in the model. First, are lexical features which count the number of times a word is assigned a particular predicate. Second, are bilexical features which count the number of times a word is assigned a particular predicate *and* a particular word precedes or follows it. Last, are rule features which count the number of times a particular composition rule is applied in the derivation. We have also extensively experimented with other features such as features including syntactic labels, and prepositional phrase features, but failed to show improvement.

The training process finds a parameter $\bar{\theta}^*$ that (approximately) maximizes the sum of the conditional log-likelihood of the MRs in the training set. Since no specific semantic derivation for an MR is provided in the training data, the conditional log-likelihood of an MR is calculated as the sum of the conditional probability of all semantic derivations that lead to the MR. Formally, given a set of NL-MR pairs $\{(S_1,M_1),(S_2,M_2),...,(S_n,M_n)\}$ and the syntactic parses of the NLs $\{T_1,T_2,...,T_n\}$, the parameter $\bar{\theta}^*$ is calculated as:

$$\begin{aligned}\bar{\theta}^* &= \arg\max_{\bar{\theta}} \sum_{i=1}^{n} \log \Pr(M_i|S_i,T_i;\bar{\theta}) \\ &= \arg\max_{\bar{\theta}} \sum_{i=1}^{n} \log \sum_{D_i^*} \Pr(D_i^*|S_i,T_i;\bar{\theta})\end{aligned} \tag{5.5}$$

where D_i^* is a semantic derivation that produces the correct MR M_i.

L-BFGS (Nocedal, 1980) is used to estimate the parameters $\bar{\theta}^*$. The estimation requires statistics that depend on all possible semantic derivations and all correct semantic derivations of an example, which are not feasibly enumerated. A variant of the Inside-Outside algorithm (Miyao and Tsujii, 2002) is used to efficiently collect the necessary statistics. Following Wong and Mooney (2006), only candidate predicates and composition rules that are used in the best semantic derivations for the training set are retained for testing. No smoothing is used to regularize the model, although we have tried using a Gaussian prior (Chen and Rosenfeld, 1999), which failed to improve the results.

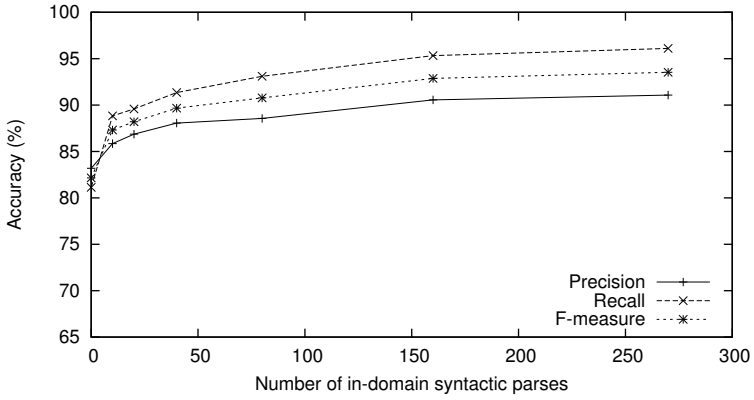
5.7 Experimental Evaluation

5.7.1 Methodology

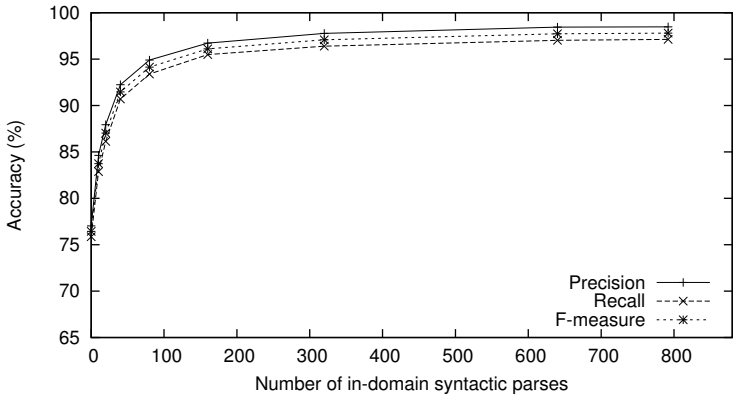
We experimented with SYNSEM on both the CLANG and GEOQUERY corpora (Section 2.3). Since in GEOQUERY, the FUNQL MRL has been shown to be less effective than the Prolog-based MRL (Section 3.6), the Prolog language was exclusively used for the GEOQUERY experiments. We shall also give results on the small GEOQUERY corpus containing 250 examples, GEO250.

The semantic parsers were evaluated using standard 10-fold cross validation, and their performance was measured in terms of *precision*, *recall*, and *F-measure* as in Section 3.6. No partial credit was given for examples with partially-correct SAPTs.

Collins (1997) parsing model 2 (Bikel, 2004) was trained on all sections of the WSJ corpus of Penn Treebank (Marcus et al., 1993) to get automated syntactic parses. The performance of a syntactic parser trained only on the WSJ corpus can degrade dramatically in new domains due to corpus variation (Gildea, 2001). Ex-



(a) CLANG



(b) GEOQUERY

Figure 5.13: Learning curves of the Bikel (2004) syntactic parser on CLANG and GEOQUERY trained on the WSJ plus a small number of in-domain examples.

periments on CLANG and GEOQUERY showed that the performance can be greatly improved by adding a small number of treebanked examples from the corresponding training set together with the WSJ. Figure 5.13 shows the results for the Bikel parser trained on an increasing amount of in-domain data, demonstrating the improved results of using a small number of in-domain sentences. We also experimented with assigning higher weight to in-domain sentences, but failed to get improvement. In the experiments, our semantic parser was evaluated using three kinds of syntactic parses, listed together with their PARSEVAL F-measures compared to the gold-standard syntactic parses annotated for SCISSOR: gold-standard parses from the treebank (GoldSyn, 100%), the Bikel parser trained on WSJ plus a small number of in-domain training sentences required to achieve good performance, 20 for CLANG (Syn20, 88.21%) and 40 for GEOQUERY (Syn40, 91.46%), and the Bikel parser trained on no in-domain data (Syn0, 82.15% for CLANG and 76.44% for GEOQUERY).

For semantic parsing, we compared SYNSEM with SCISSOR, WASP, KRISP, and LU on CLANG, and λ-WASP and Z&C on GEOQUERY which also used the Prolog MRL. The details of these systems and their variances have been shown in Section 3.6.1, briefly:

- SCISSOR (Chapter 3) is an integrated syntactic-semantic parser, which requires annotated SAPTs for training.
- WASP (Wong and Mooney, 2006) is a semantic parser based on machine translation techniques.
- λ-WASP (Wong and Mooney, 2007) is an extension of WASP for handing logical forms.

	FULL SIZE (270)			SMALL SIZE (40)		
	Precision	Recall	F-measure	Precision	Recall	F-measure
GOLDSYN	84.73	74.00	79.00	61.14	35.67	**45.05**
SYN20	85.37	70.00	76.92	57.76	31.00	40.35
SYN0	87.01	67.00	75.71	53.54	22.67	31.85
SCISSOR	89.50	73.70	**80.80**	85.00	23.00	36.20
WASP	88.85	61.93	72.99	88.00	14.37	24.71
KRISP	85.20	61.85	71.67	68.35	20.00	30.95
LU	82.50	67.70	74.40	–	–	–

Table 5.3: Performance on CLANG, using all training examples and a small amount of them.

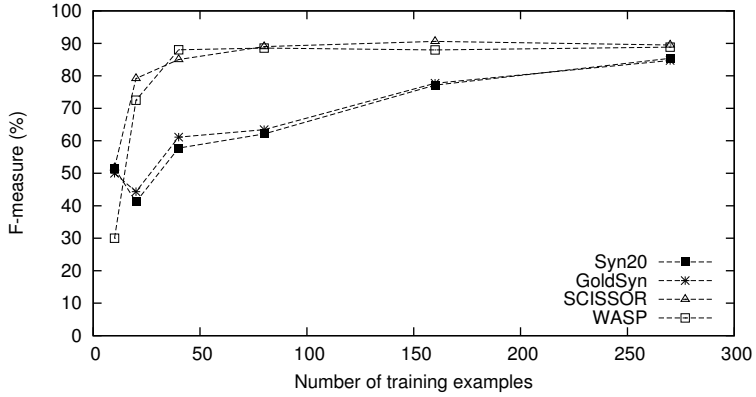
- KRISP (Kate and Mooney, 2006) is a semantic parser based on string kernels.
- Z&C (Zettlemoyer and Collins, 2007) is a probabilistic semantic parser using CCG, which requires a set of hand-built grammar template rules for the specific NL as prior knowledge.
- LU (Lu et al., 2008) is a generative semantic parsing model, which uses discriminative reranking.

5.7.2 Results

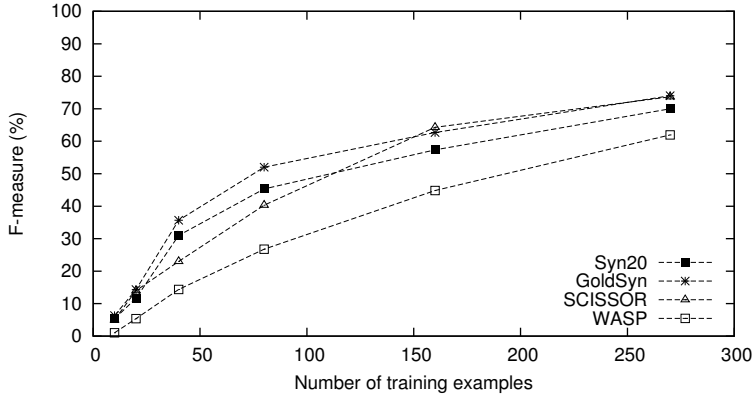
5.7.2.1 Results on CLANG and Discussions

Table 5.3 summarizes the performance of the semantic parsers on CLANG using all training data (270 examples) and a small amount of them (40) respectively[4]. Figure 5.14 shows the available learning curves, where SYN0 and KRISP are ommitted for clarity. Several observations can be made:

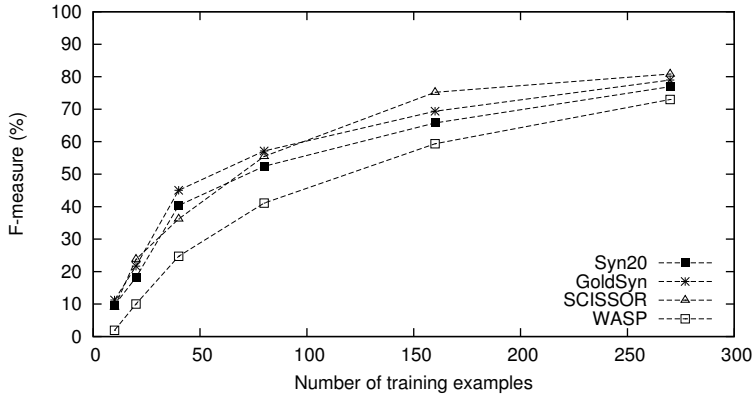
[4]Lu et al. (2008)'s F-measure on CLANG before reranking is 67.8%.



(a) Precision



(b) Recall



(c) F-measure

Figure 5.14: Learning curves for semantic parsers on CLANG.

- More accurate syntactic parsers (GOLDSYN > SYN20 > SYN0) improved this approach. Although it may sound obvious, empirically, it confirmed our fundamental assumption of this thesis that overall, syntactic parses provide the correct structure for meaning composition.

- When using all training data, all SYNSEM parsers outperformed all other systems except SCISSOR which requires extra SAPT annotation. This suggests that SYNSEM can leverage accurate syntactic parsers to produce accurate semantic parsers.

- When using a small amount of training data, both GOLDSYN and SYN20 outperformed WASP and KRISP substantially; they even outperformed SCISSOR which requires extra annotation. This shows that SYNSEM significantly improves results when limited training data is available. This demonstrates the advantage of utilizing an accurate existing syntactic parser, where syntactic structure is learned from large open domain treebanks instead of relying just on the training data.

- An additional point with limited training data is that SYN0 failed to show great improvement over WASP and KRISP by using an existing parser. As we can see, reducing the training data from 270 to 40 increased the differences among these parsers. This suggests that the quality of syntactic parsing becomes critically important when limited training data is available.

The F-measures of syntactic parses that generated correct MRs on CLANG were 85.50% in SYN0 and 91.16% in SYN20, which suggests that SYNSEM can parse on imperfect syntactic parses, due to its ability to ensure meaning composition on imperfect syntactic parses during training (Section 5.3). For example, one

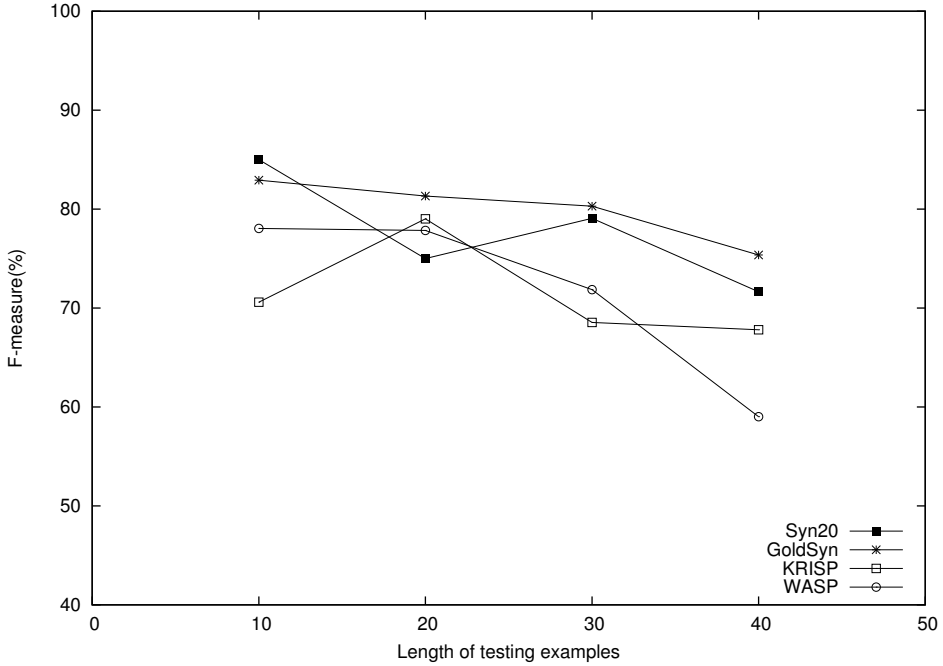
syntactic error that SYN20 successfully generated the MR for was the conjunction error in the phrase *players 2,3,7 and 8*, where *7 and 8* were mistakenly separated from *player*, and instead attached to the following phrase. Even though a syntactic parser can be not accurate enough for learning from limited training data, when given sufficent training data, a semantic parser based on it can still learn successfully (SYN0). However, we note that a rationally accurate syntactic parser should be used for SYNSEM. When evaluating SYNSEM using a syntactic parser trained only on Section 1 of the WSJ (PARSEVAL F-measure = 59.2%), its semantic F-measure dropped dramatically to about 46%.

A detailed analysis on CLANG (see Figure 5.15) shows that our improved performance on CLANG compared to WASP and KRISP was mainly for long sentences (> 20 words). Utilizing syntactic knowledge from an existing parser provides the predicate-argument structure for compositional semantic analysis, which can be hard for the semantic-grammar-based parsers to learn when sentences are long. Similarly, utilizing syntactic knowledge learned from the SAPT annotation in SCISSOR also successfully guides the correct compositional semantic analysis (see Figure 3.11).

5.7.2.2 Results on GEOQUERY and Discussions

Table 5.4 summarizes the performance of the semantic parsers on GEOQUERY using all training examples, and Figure 5.16 shows the available learning curves, where SYN0 is ommited for clarity. Several observations can be made:

- SYN0 performed significantly worse than λ-WASP and other SYNSEM parsers using more accurate syntactic parses. This is not surprising since SYN0's F-measure for syntactic parsing was only 76.44% in GEOQUERY due to a lack



(a) Results

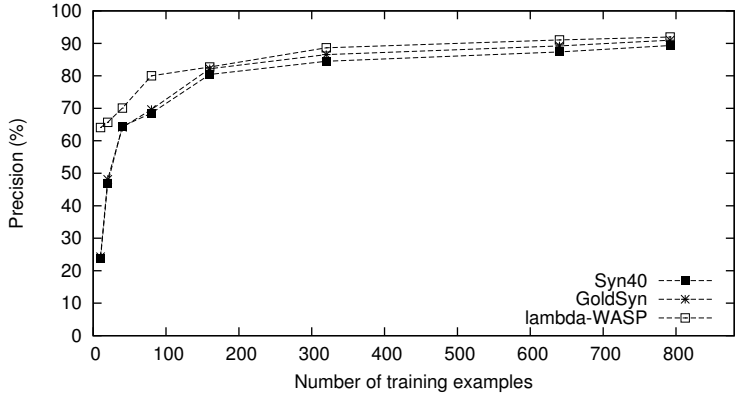
NL length	0-10	11-20	21-30	31-40	41-50
Sentence count	22	98	137	38	5
Corresponding point in the figure above	*10*	*20*	*30*	*40*	–

(b) NL-length distribution

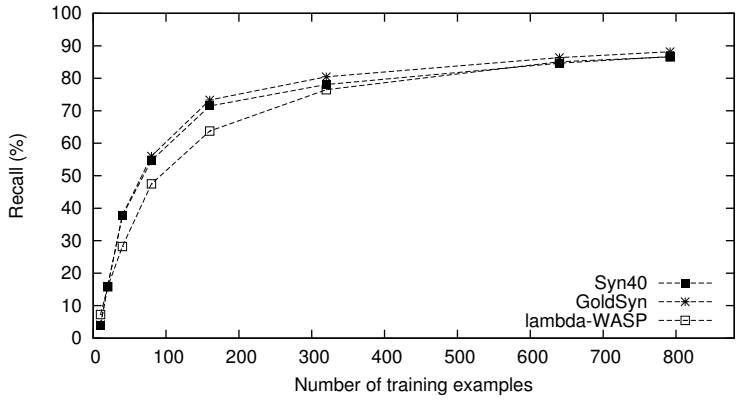
Figure 5.15: CLANG results on test sentences within length ranges, where most sentences are within the ranges of $11-20$ and $21-30$.

	Precision	Recall	F-measure
GOLDSYN	91.94	88.18	**90.02**
SYN40	90.21	86.93	88.54
SYN0	81.76	78.98	80.35
λ-WASP	91.95	86.59	89.19
Z&C	91.63	86.07	88.76

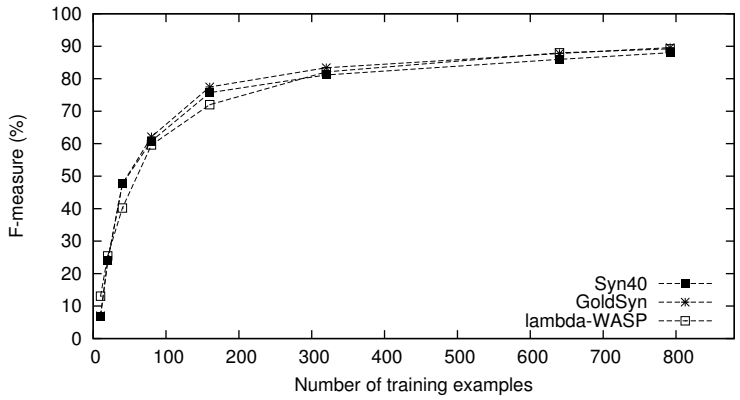
Table 5.4: Performance on GEOQUERY.



(a) Precision



(b) Recall



(c) F-measure

Figure 5.16: Learning curves for semantic parsers on GEOQUERY.

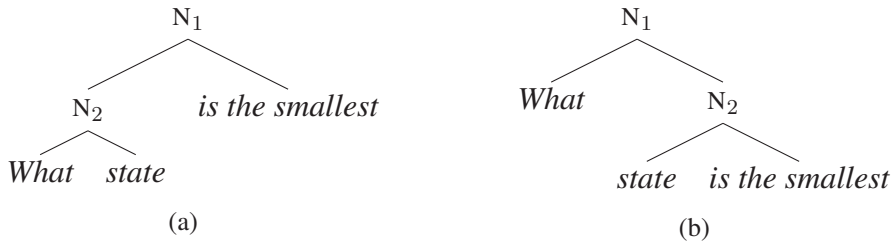


Figure 5.17: A simple example to illustrate the limitation of SYNSEM and SCISSOR. (a) The sentence structure given in SYNSEM and SCISSOR. (b) A possible sentence structure learned by a semantic grammar.

of interrogative sentences (questions) in the WSJ corpus. This suggests that a rationally accurate syntactic parser should be used for SYNSEM.

- All of SYN40, GOLDSYN and λ-WASP performed highly competitively all across the learning curves. This could be explained by the characteristic of GEOQUERY where sentences are generally short (7.57 words on average). When sentences are short, utilizing syntactic parses does not necessarily benefit semantic parsing by providing meaning composition structure; the difficulty in learning semantic knowledge (SYNSEM) can approximately be equal to the difficulty in learning semantic knowledge together with syntactic knowledge (λ-WASP). However, when sentences are long (CLANG), syntactic parses can provide the basic predicate-argument structure which can be hard to learn. This is consistent with our observation on CLANG.

In fact, when sentences are short, utilizing the predefined syntactic structure in a syntactic parse or SAPT (Chapter 3) is sometimes less flexible than learning directly from the NL sentences. Consider the following example:

	Precision	Recall	F-measure
GOLDSYN	95.73	89.60	92.56
SYN20	93.19	87.60	90.31
SYN0	91.81	85.20	88.38
λ-WASP	91.76	75.60	82.90

Table 5.5: Performance on GEO250 (20 in-domain sentences are used in SYN20 to train the syntactic parser).

What state is the smallest?

`answer(`x_1`,smallest(`x_1`,state(`x_1`)))`

A syntactic parser would typically produce the sentence structure in Figure 5.17(a), where the argument `state` is attached to its ancestor predicate `answer`, which is non-isomorphic to its MR parse. While a parser based on semantic grammars such as λ-WASP would learn the structure in Figure 5.17(b), where the argument `state` is isomorphically attached to the predicate `smallest` due to its flexibility in learning directly from the NL strings . As a result, the non-isomorphic syntactic structure introduced by a predefined syntax can increase the difficulty of learning semantic parsers, particularly when using an inaccurate syntactic parser. We note that Zettlemoyer and Collins (2005, 2007) also adopt a syntax-based approach. Unlike SYNSEM and SCISSOR, it does not rely on a predefined syntax, thus it bears the same flexibility as the approaches based on semantic grammars.

Performance of available semantic parsers on GEO250 using the Prolog MRL is shown in Table 5.5. Similar to the performance of SCISSOR which also explores the knowledge of syntax, all SYNSEM parsers significantly outperformed λ-WASP. Again, it would be interesting to investigate the linguistic differences between GEOQUERY and GEO250 which may cause the performance difference, since GEOQUERY was collected from more diverse resources than GEO250 (Sec-

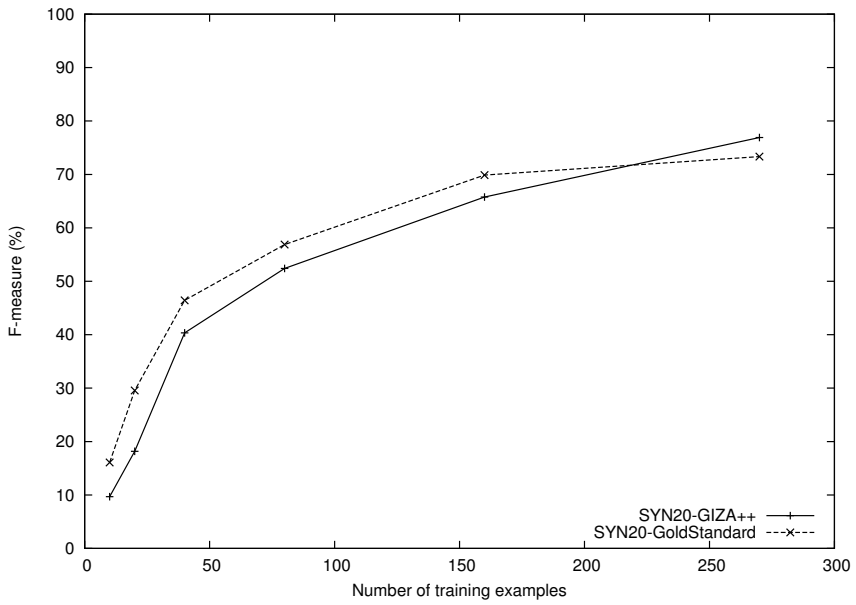


Figure 5.18: Learning curves of Syn20 on CLang utilizing GIZA++ and gold-standard word alignment models.

tion 2.3).

5.7.2.3 Other Results

We also evaluated the impact of the word alignment component by replacing Giza++ by gold-standard word alignments manually annotated for the CLANG corpus (See Figure 5.18). The results consistently showed that compared to using gold-standard word alignment, Giza++ produced lower semantic parsing accuracy when given little training data, but better results when given sufficient training data ($= 270$ examples). This suggests that, given sufficient data, Giza++ can produce effective word alignments, and that imperfect word alignments do not seriously impair our semantic parsers since the disambiguation model evaluates multiple possi-

ble interpretations of ambiguous words. Using multiple potential alignments from Giza++ sometimes allows for even better performance than using a single gold-standard word alignment because it permits multiple interpretations to be evaluated by the global disambiguation model.

5.7.2.4 Summary

The observations are summarized as follows:

- SYNSEM can produce accurate semantic interpretations by utilizing an accurate syntactic parser. On CLANG, SYNSEM outperformed all other systems except SCISSOR which requires extra SAPT annotation. On GEOQUERY, when utilizing accurate syntactic parses, SYNSEM is competitive to other semantic parsers.
- SYNSEM significantly improves results with limited training data by using an accurate syntactic parser to provide syntactic knowledge; the quality of syntactic parsing becomes critically important with limited training data.
- SYNSEM significantly improves results on long sentences when syntactic parses can provide the basic predicate-argument structure which is hard to learn for parsers based on semantic grammars.
- When sentences are short, utilizing the syntactic structure predefined in a syntactic parse or SAPT (Chapter 3) is sometimes less flexible than learning directly from the NL sentences.
- SYNSEM is robust to syntactic errors.

We have shown that by utilizing a syntax-driven approach, SYNSEM is capable of learning accurate semantic parsers based on automated syntactic parsers.

On the other hand, we note that automated syntactic parses have also been used in the semantic-grammar-based parsers to learn semantic production rules (see Section 2.4.3). SILT (Kate et al., 2005) learns a tree version of a semantic grammar where the production rules are generalized from automated syntactic parses. Its F-measure on CLANG is 64%, which is lower than that of SYN0, 75.71%. KRISP (Kate, 2007) utilizes tree kernels instead of string kernels to learn a semantic grammar. No results were reported on CLANG. On the GEOQUERY using FUNQL, its performance using both gold-standard and automated syntactic parses were lower than that of the parser learned directly from the NL strings. Thus, it failed to demonstrate the same strengths of utilizing syntactic parses as in SYNSEM.

5.8 Increasing Robustness to Syntactic Parsing Errors

Our SYNSEM semantic parsing algorithm uses the best parse trees from an existing syntactic parser to drive the interpretation process. Hence, if a syntactic parse has significant errors affecting semantic construction, the correct MR may not be generated. For example, if the word *2* in Figure 5.3(c) is mistakenly attached to *has* instead of *player*, the predicate-argument relation between the player and its number cannot be analyzed. We note that in training, to be robust to syntactic errors, our algorithm learns to construct correct MRs even from syntactic parses with errors. In testing, however, the algorithm may fail if syntactic errors have not been seen. In this section, we present our work on increasing robustness to syntactic errors by utilizing the k-best syntactic parses produced by a syntactic parser.

A similar problem is also faced by a related task of semantic role labeling (SRL) where semantic role identification and classification are only applied to nodes (constituents) in a syntactic parse tree. Thus, if a target word's argument does not have its corresponding syntactic node constructed correctly, it will not be

k	1	2	5	10
F-measure	88.21	89.04	89.37	89.43

Table 5.6: the oracle F-measure of the syntactic parser used in SYN20 on CLANG.

identified. Much work has been done to address this problem by utilizing the k-best syntactic parses (Gildea and Jurafsky, 2002; Sutton and McCallum, 2005; Haghighi et al., 2005), where a syntactic parser is used to generate the k-best syntactic parses for an example, and a base SRL system is then used to generate an SRL for each parse. The output SRL is mainly chosen to be the one with the best score: Gildea and Jurafsky (2002), and Haghighi et al. (2005) use simple ways to combine the scores from the syntactic parser and the SRL system, while Sutton and McCallum (2005) also learn a sophisticated reranking system to rank the candidate SRLs. All approaches fail to show improvement on this task.

In our experiments, we adopted a similar simple methodology for utilizing multiple syntactic parses in testing. First, Bikel's implementation of Collins' parsing model (Bikel, 2004) is used to generate the best k syntactic parses for each test example (beam width = 10^4). Second, the learned SYNSEM algorithm is used to generate an MR for each syntactic parse, starting from the best until a complete MR is constructed. Finally, the complete MR is returned as the MR of the example. For experimental setting, we applied the semantic parser SYN20 to CLANG.

Generally, more accurate syntactic parses generate more accurate MRs (cf. Section 5.7). Thus, to decide k, we first measured the oracle performance of the syntactic parser used in SYN20, where the oracle picks the syntactic parse with the highest F-measure among the k parses. If utilizing more syntactic parses do not improve the oracle syntactic parser, it may also fail to improve the semantic parser. Table 5.6 summarizes the oracle F-measure on CLANG as a function of number k,

	Precision	Recall	F-measure
SYN20	85.37	70.00	76.92
SYN20+	85.65	71.67	78.04

Table 5.7: Performance of the augmented semantic parser SYN20+ on CLANG (Oracle recall: 72.00).

It shows that increasing k above 5 only slightly improves the syntactic parser, thus k is set to be 10 in our experiment, where 278 (total 300) examples have their oracle parses scored the highest by the original syntactic parser.

Table 5.7 shows the performance of the semantic parser Syn20 augmented with the simple approach previously proposed (Syn20+). The results show a slight, but not statistically significant (based on paired t-test), increase in the performance. To better understand the results, we measured the oracle recall of SYN20+ (k = 10) where if any of the MRs constructed from the syntactic parses were correct, the example was also correct. This is the optimal recall any reranking approach can achieve using the same base semantic parser. It is found that the recall achieved by the simple approach (71.67%) is very close to the oracle recall (72%), thus more sophisticated reranking approaches will not help. Surprisingly, the possible improvement between this oracle recall and the recall using the top one syntactic parse (70%) is only 2%.

Besides reranking the multiple parses from a single syntactic parser, other approaches have also been employed to reduce the effect of syntactic errors. In semantic parsing, Popescu et al. (2004) utilize a hand-crafted semantic lexicon and a set of semantic constraints to correct syntactic errors of a syntactic parse. In SRL, Pradhan et al. (2005b) and Koomen et al. (2005) perform fine granuality combination from semantic roles generated using multiple syntactic outputs. We are also interested in utilizing multiple alternative syntactic parses in training SYNSEM.

5.9 Conclusion

We have presented a new approach to learning a semantic parser that utilizes an existing syntactic parser to drive compositional semantic interpretation. By exploiting an existing syntactic parser trained on a large treebank, our approach produces improved results on standard corpora, particularly when training data is limited or sentences are long. The approach also exploits methods from statistical MT (word alignment) and therefore integrates techniques from statistical syntactic parsing, MT, and compositional semantics to produce an effective semantic parser.

Chapter 6

Future Work

In this chapter, we discuss several future research directions for this thesis.

6.1 Improving SCISSOR and SYNSEM

There are a number of ways in which SCISSOR and SYNSEM can be improved. SCISSOR uses a generative model for semantic parsing, which has a limited choice of features due to its generative nature. In Chapter 4, we have experimented with improving its performance by using discriminative reranking, which explores arbitrary global features. In the future, we would like to experiment with discriminative methods (Finkel et al., 2008) for semantic parsing, which have more flexibility in selecting features.

Another potential improvement of SCISSOR is to extend it for handling MRLs with logical variables. In the current SCISSOR framework, labels in SAPTs are the combination of syntactic and semantic labels. Thus it would be hard to directly incorporate logical variables in semantic labels, since further adding logical variables into semantic labels would make the model more complex. Alternatively, the unification knowledge of logical variables can be learned in the process that transforms SAPTs to MRs by learning composition rules similar to those in SYNSEM.

The integrated approach in SCISSOR allows semantic information to be

available during parsing time, so that the parser can find a globally most likely parse for both syntactic and semantic interpretation. However, it requires the extra annotation of SAPTs. Thus, another interesting area of future research would be to explore an integrated parser like SCISSOR, that does not require the extra annotation. In such a parser, syntax can be learned in an unsupervised manner (Klein and Manning, 2002, 2004) under the supervision of semantic knowledge. Ideally semantics would aid the unsupervised grammar induction. On the other hand, the semantic parsers can also provide possible evaluation for unsupervised syntactic parsers that is more indicative of quality.

SYNSEM exploits an existing syntactic parser to produce disambiguated parse trees that drive the compositional semantic interpretation, thus, it has the advantage of conveniently leveraging the progress in syntactic parsing. In this thesis, we utilized the Collins parser so that its results were comparable to those of SCISSOR. In the future, we would like to experiment with syntactic parsers with improved accuracy (Charniak and Johnson, 2005; Huang, 2008) and using diverse grammar formalism (Riezler et al., 2002; Clark and Curran, 2004). Particularly, we would like to experiment with dependency parsers (McDonald et al., 2005; Koo et al., 2008) which focus on modeling word dependencies rather than phrase structure, which might be more suitable for modeling predicate-argument relations in semantic parsing. We would also like to experiment with using many syntactic parsers in parallel, possibly informing the same decisions (Pradhan et al., 2005b; Koomen et al., 2005).

6.2 Semantic Parsing Utilizing Wide-coverage Semantic Representations

SYNSEM exploits an existing syntactic parser to produce disambiguated

parse trees that drive the compositional semantic interpretation. When based on syntactic parse trees, SYNSEM needs to learn knowledge for syntactic variations that represent the same meaning. For example, the following examples have varied syntax:

Pass the ball to John.
The ball is passed to John.

for the same meaning[1]:

x_0, ball(x_0)
x_1, named(x_1, john, per)
x_2, {pass(x_2), event(x_2), patient(x_2, x_0), to(x_2,x_1)}

Instead of relying on syntactic parses, a semantic parser can alternatively rely on wide-coverage meaning representations generated from syntactic parses, but with syntactic variations normalized, to reduce the complexity of handling syntactic variations in semantic parsing. In such a parser, the system must learn to construct the wide-coverage meaning representations to the MRs in a target MRL.

Combinatory Categorial Grammar (CCG) (Steedman, 2000) has an appealing integrated treatment of syntax and semantics, in which linguistic phenomena including long dependencies are well addressed. By using CCG, wide-coverage meaning representations can be practically generated from syntactic parses (Hockenmaier et al., 2004; Curran et al., 2007). Boxer (Curran et al., 2007) is such a wide-coverage compositional semantic tool that takes a CCG derivation output from a high performance CCG parser (Clark and Curran, 2004), and generates a

[1]Generated using the Boxer demo at http://svn.ask.it.usyd.edu.au/trac/candc/wiki/Demo. The MR in Discourse Representation Theory is not strictly shown for brevity.

semantic meaning representation with syntactic variances removed, including long-dependency variations as illustrated in the following example:

the food that John cooked
x_0, food(x_0)
x_1, named(x_1, john, per)
x_2, {cook(x_2), event(x_2), agent(x_2, x_1), patient(x_2, x_0)}

Note systems like Boxer do not handle word senses or map words to a formal ontology.

By using wide-coverage meaning representations as input instead of NLs or syntactic parses, semantic parsers such as WASP and SYNSEM can reduce the complexity in learning syntactic variations, and therefore hopefully learn more efficiently.

6.3 Utilizing Semantic Role Labeling for Semantic Parsing

Semantic role labeling (SRL) (Gildea and Palmer, 2002) is the task of identifying target words and their semantic roles in natural language sentences, which is an important task toward natural language understanding beyond syntactic parsing. With the availability of the large open-domain annotated SRL corpora such as FrameNet (Fillmore et al., 2002), PropBank (Palmer et al., 2005), and OntoNotes (Hovy et al., 2006), semantic role labeling has made fast progress (Carreras and Marquez, 2004, 2005) in recent years. Consider the following example with the PropBank style SRL annotation paired with its MR in CLANG[2]:

[2]The semantic roles were generated using the SRL demo at http://l2r.cs.uiuc.edu/ cogcomp/srl-demo.php.

[$_{A0}$*Player 2*] [$_{V}$*passes*] [$_{A1}$*the ball*] [$_{A2}$*to player 3*]

```
(do (player our {2}) (pass (player our {3})))
```

where the target verb is *passes*, the semantic role A_0 means the giver, A_1 means the things given, and A_2 means the entity given to. Although the outputs of SRL are not complete and directly executable, they still extract interesting semantic relations between target words and their semantic roles, which often correlate with the predicate-argument relations in semantic parsing. For example, in the example above, the semantic role A_2 of the target verb *passes* correlates with the only argument of the predicate P_PASS in the MR.

Thus, beyond syntactic parsing, semantic role labeling can provide a layer of semantic information for semantic parsing. It would be interesting to investigate possible ways to leverage SRL in semantic parsing. One direction could be to provide discriminative features for semantic parsing. For example, in SYNSEM, when composing the meanings of *passes the ball* and *to player 3*, since in the SRL, the phrase *to player 3* is the recipient of the word *passes*, it could suggest that in the MR, it should also fill the only argument of the predicate P_PASS (*passes*). Another direction could be to provide improved generalization for semantic parsing. For example, if we see the following sentence in testing:

[$_{A0}$*Player 2*] [$_{V}$*passes*] [$_{A1}$*the ball*] [$_{A2}$*to the goalie*].

where *the goalie* is not seen as a player in training, since in the SRL, *to the goalie* also fills *passes*' semantic role A_2 as in the previous example, we could confidently conclude that in the MR, it also represents a player which fills the only argument of the predicate P_PASS.

6.4 Open-domain Natural Language Understanding

Until now, most work on deep natural language understanding has focused on a specific domain (Wong and Mooney, 2006; Kate and Mooney, 2006; Zettlemoyer and Collins, 2007), where domain knowledge can be represented in a meaning representation language; while open-domain natural language understanding has mainly focused on relatively shallow tasks such as information extraction and semantic role labeling.

A major challenge to open-domain semantic parsing is that its intricacy makes it difficult, if not possible, to design a global meaning representation language for open domains. For example, what should be the concepts and their semantic relations? In response to this, Poon and Domingos (2009) propose unsupervised semantic parsing which avoids the difficulty of defining an open-domain MRL, assuming that the concepts in open domains can be viewed as clusters of syntactic or lexical variations of the same meaning, and their relations can be defined by their syntactic relations. For example, the concept of C_BUY can be viewed as the cluster of *buys, acquires, 's purchase of*, as in the following sentences:

Microsoft buys PowerSet
Microsoft acquires PowerSet
Microsoft's purchase of PowerSet

MacCartney and Manning (2008) propose utilizing natural logic which supports directly reasoning in natural language.

These two approaches represent interesting work on open-domain natural language understanding. However, they largely ignored the rich open-domain knowledge base currently available, such as WordNet (Fellbaum, 1998), which roughly defines the concepts in open domains, and PropBank (Palmer et al., 2005)

and OntoNotes (Hovy et al., 2006), which roughly define the semantic relations among the concepts. There have been research efforts aimed at integrating these concepts and relation knowledge bases, such as in Omega (Philpot et al., 2005). Moreover, Wikipedia can be used to provide a large, constantly evolving ontology (Strube and Ponzetto, 2007). In future work, we would like to explore the possibilities of leveraging these open-domain knowledge base for deep language understanding. For example, we would like to see if the knowledge base can provide disambiguation knowledge and expectations for concept clustering in Poon and Domingos (2009), or if the knowledge learned for one lexical word or concept can be propagated to other words or concepts according to their relations in the knowledge base, so that we can learn efficiently. In turn, the knowledge learned in these systems can also be used to enrich the knowledge base, which forms an integrated growth cycle of knowledge base and natural language understanding. This interesting idea has been investigated in Barker et al. (2007).

Chapter 7

Conclusion

Natural language understanding is a fundamental problem in natural language processing in terms of its theoretical and empirical importance. In recent years, startling progress has been made at different levels of natural language processing tasks which provides great opportunity for deeper natural language understanding. In this thesis, we focused on semantic parsing, which maps a natural language sentence into a complete, formal meaning representation in a meaning representation language. We presented two novel learned syntax-based semantic parsers using statistical syntactic parsing techniques, motivated by the following two reasons. First, the syntax-based semantic parsing is theoretically well-founded in computational semantics. Second, adopting a syntax-based approach allows us to directly leverage the enormous progress made in statistical syntactic parsing.

We first introduced SCISSOR (Chapter 3), an integrated syntactic-semantic parsing approach, in which the Collins (1997) syntactic parser is augmented with semantic parameters to produce a semantically-augmented parse tree; this tree is then translated into a final formal meaning representation. This integrated approach allows semantic information to be available during parsing time, so that the parser can find a globally most likely parse for both syntactic and semantic interpretation to obtain an accurate combined syntactic-semantic analysis. Training SCISSOR required that sentences be annotated with SAPTs as well as MRs. We reported experimental results on two real applications, an interpreter for coaching

instructions in robotic soccer (CLANG) and a natural-language database interface (GEOQUERY). On CLANG, SCISSOR substantially outperformed all other systems; on GEOQUERY, it was comparable to all other semantic parsers except the recent Lu et al. (2008). Analysis showed that the main improvement of SCISSOR over other systems was on long sentences, where the annotated SAPTs provided the knowledge of accurate meaning composition structure; while it is hard for other approaches to infer the syntactic knowledge directly from sentences only paired with MRs when sentences are long.

The generative model in SCISSOR has a limited choice of features due to its generative nature, and its performance can potentially be improved by using discriminative reranking, which explores arbitrary global features. In chapter 4, we investigated discriminative reranking upon SCISSOR, examining if global features used for reranking syntactic parsing can be adapted for semantic parsing by creating similar semantic features based on the mapping between syntax and semantics. We reported experimental results on CLANG and GEOQUERY, showing that reranking improved the performance on CLANG but not on GEOQUERY, where sentences are short which are less likely for global features to show improvement on.

We then introduced the second semantic parser called SYNSEM (Chapter 5), which does not require the SAPT annotation as in SCISSOR, but exploits an existing syntactic parser instead to produce disambiguated parse trees that drive the compositional semantic interpretation. With the advancement of statistical syntactic parsing, accurate syntactic parsers are available for many languages and could potentially be used to learn more effective semantic analyzers. Thus, this pipeline approach allows semantic parsing to conveniently leverage the progress in syntactic parsing. We reported experimental results on CLANG and GEOQUERY. On CLANG, SYNSEM outperformed all other systems except SCISSOR which requires

extra SAPT annotation; on GEOQUERY, when utilizing accurate syntactic parses, SYNSEM is competitive to all other semantic parsers. Analysis showed that SYNSEM significantly improved results with limited training data by using an accurate syntactic parser providing syntactic knowledge; similarly to SCISSOR, it also improved results on long sentences due to the prior syntax knowledge. It has also been shown to be robust to syntactic errors.

In these two syntax-based approaches, the knowledge of traditional syntactic analysis comes in the form of annotated SAPTs and syntactic parses from an existing syntactic parser, incorporating the knowledge of accurate sentence structures for compositional semantic analysis. When sentences are short, the knowledge of syntax encoded in SCISSOR and SYNSEM can feasibly be learned directly from NL sentences paired with their MRs, as illustrated in the GEOQUERY domain. In fact, the approaches utilizing non-syntactic knowledge (Zettlemoyer and Collins, 2005; Wong and Mooney, 2006; Kate and Mooney, 2006) can sometimes be even more flexible in exploring optional feasible sentence structures for meaning composition, and in finding the best one which is more isomorphic to the underlying MR structure. However, when sentences are long, it gets harder for these approaches to infer the syntactic knowledge directly from sentences paired with MRs. SCISSOR and SYNSEM show their strengths here by using prior knowledge of syntactic analysis to guide the basic predicate-argument structure for meaning composition, which outweighs the lost flexibility constrained by this knowledge.

Overall, this thesis contributed in improving the task of semantic parsing: while the mainstream of the syntax-based approaches for semantic parsing is hand-built, this thesis proposed two state-of-the-art learned semantic parsers based on syntax which leverage both the techniques and results of the enormous progress made in statistical syntactic parsing. We showed that the main improvement of

SCISSOR and SYNSEM over other systems is on long sentences, where the prior syntactic knowledge given in the form of annotated SAPTs or syntactic parses from an existing parser helps semantic composition. When comparing SCISSOR and SYNSEM, we showed that SCISSOR outperformed SYNSEM when given sufficient training data, by utilizing the annotated SAPTs. However, when given limited training data, SYNSEM gave the best state-of-the-art results by using an accurate syntactic parser as syntactic knowledge.

Bibliography

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.

Jason Baldridge, Sudipta Chatterjee, Alexis Palmer, and Ben Wing (2007). Dotccg and visccg: Wiki and programming paradigms for improved grammar engineering with openccg. In *Proceedings of the Workshop on Grammar Engineering Across Frameworks*. Stanford, CA.

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 Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, pp. 280–286. Vancouver, Canada.

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.

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

Johan Bos (2006). Three stories on automated reasoning for natural language understanding. In *Proceedings of the ESCoR: Successful Computerized Reasoning*, pp. 81–91. Seattle, WA.

Johan Bos, Bjorn Gamback, Christian Lieske, Yoshiki Mori, Manfred Pinkal and Karsten Worm (1994). Compositional semantics in verbmobil. In *Proceedings of the Sixteenth International Conference on Computational Linguistics*. Copenhagen, Denmark.

Razvan C. Bunescu and Raymond J. Mooney (2005). A shortest path dependency kernel for relation extraction. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-05)*, pp. 724–731. Vancouver, BC.

Mary Elaine Califf and Raymond J. Mooney (1999). Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pp. 328–334. Orlando, FL.

Xavier Carreras, Michael Collins and Terry Koo (2008). Tag, dynamic programming and the perceptron for efficient, feature-rich parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL-2008)*, pp. 9–16. Manchester,UK.

Xavier Carreras and Luis Marquez (2004). Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*. Boston, MA.

Xavier Carreras and Luis Marquez (2005). Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Com-*

putational Natural Language Learning (CoNLL-2005), pp. 152–164. Ann Arbor, MI.

Eugene Charniak (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pp. 598–603. Providence, RI.

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

Eugene Charniak and Mark Johnson (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pp. 173–180. Ann Arbor, MI.

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.

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-04)*, pp. 104–111. Barcelona, Spain.

Michael Collins (1999). *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Michael Collins (2000). Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, pp. 175–182. Stanford, CA.

Michael Collins (2002a). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*. Philadelphia, PA.

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

Michael Collins (2002c). Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 489–496. Philadelphia, PA.

Michael Collins (2003). Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Michael Collins (2004). Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In Harry Bunt, John Carroll and Giorgio Satta, eds., *New Developments in Parsing Technology*. Kluwer.

Michael Collins, Philipp Koehn and Ivona Kucerova (2005). Discriminative syntactic language modeling for speech recognition. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pp. 507–514. Ann Arbor, MI.

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.

Brooke Cowan and Michael Collins (2005). Morphology and reranking for the statistical parsing of spanish. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-05)*. Vancouver, B.C., Canada.

James Curran, Stephen Clark and Johan Bos (2007). Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pp. 33–36. Prague, Czech Republic.

John Dowding, Jean Mark Gawron, Doug Appelt, John Bear, Lynn Cherny, Robert Moore and Doug Moran (1993). Gemini: A natural language system for spoken-language understanding. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL-93)*.

Jason Eisner (2003). Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), Companion Volume*, pp. 205–208. Sapporo.

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

Charles J. Fillmore, Baker, Collin F. and Hiroaki Sato (2002). The FrameNet database and software tools. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*, pp. 1157–1160. Las Palmas.

Jenny Rose Finkel, Alex Kleeman and Christopher D. Manning (2008). Efficient, feature-based, conditional random field parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pp. 959–967. Columbus, Ohio.

Jenny Rose Finkel and Christopher D. Manning (2009). Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT-09)*, pp. 326–334. Boulder, Colorado.

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-06)*. Sydney, Australia.

Ruifang Ge and Raymond J. Mooney (2009). Learning a compositional semantic parser using an existing syntactic parser. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pp. 611–619. Suntec,Singapore.

Daniel Gildea (2001). Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-01)*. Pittsburgh, PA.

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

Daniel Gildea and Martha Palmer (2002). The necessity of syntactic parsing for predicate argument recognition. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 239–246. Philadelphia, PA.

Adele E. Goldberg (1995). *Constructions: A Construction Grammar Approach to Argument Structure*. University Of Chicago Press, Chicago, IL.

Sharon Goldwater and Tom Griffiths (2007). A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, pp. 744–751. Prague, Czech Republic.

Aria Haghighi, Kristina Toutanova and Christopher D. Manning (2005). A joint model for semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005) Shared Task on Semantic Role Labeling*. Ann Arbor, MI.

Yulan He and Steve Young (2005). Semantic processing using the hidden vector state model. *Computer Speech and Language*, 19(2):85–106.

Gary G. Hendrix, Earl D. Sacerdoti, Daniel Sagalowicz and Jonathan Slocum (1978). Developing a natural language interface to complex data. *ACM Transactions on Database Systems*, 3(2):105–147.

Julia Hockenmaier, Gann Bierner and Jason Baldridge (2004). Extending the coverage of a ccg system. In *Research in Language and Computation 2*, pp. 165–208. springer, Netherlands.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw and Ralph Weischedel (2006). OntoNotes: The 90% solution. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL-06)*. New York, NY.

Liang Huang (2008). Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pp. 586–594. Columbus, OH.

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

Nancy A. Ide and Jean Jéronis (1998). Introduction to the special issue on word sense disambiguation: The state of the art. *Computational Linguistics*, 24(1):1–40.

Thorsten Joachims (2002). Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*. Edmonton, Canada.

Rohit J. Kate (2007). *Learning for Semantic Parsing with Kernels under Various Forms of Supervision*. Ph.D. thesis, Department of Computer Sciences, University of Texas, Austin, TX.

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-06)*, pp. 913–920. Sydney, Australia.

Rohit J. Kate, Yuk Wah Wong and R. J. Mooney (2005). Learning to transform natural to formal languages. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pp. 1062–1068. Pittsburgh, PA.

Dan Klein and Chris Manning (2002). A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*. Philadelphia, PA.

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

Terry Koo, Xavier Carreras and Michael Collins (2008). Simple semi-supervised dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pp. 595–603. Columbus,OH.

Peter Koomen, Vasin Punyakanok, Dan Roth and Wen-tau Yih (2005). Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pp. 181–184. Ann Arbor, MI.

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.

Iddo Lev, Bill MacCartney, Christopher D. Manning and Roger Levy (2004). Solving logic puzzles: From robust processing to precise semantics. In *Proceedings of the Second Workshop on Text Meaning and Interpretation, ACL-04*. Barcelona, Spain.

Chang Liu, Hui Wang, Sally Mcclean, Jun Liu and Shengli Wu (2007). Syntactic information retrieval. In *Proceedings of the 2007 IEEE International Conference on Granular Computing*. Silicon Valley, CA.

Wei Lu, Hwee Tou Ng, Wee Sun Lee and Luke S. Zettlemoyer (2008). A generative model for parsing natural language to meaning representations. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP-08)*. Honolulu, Hawaii.

Bill MacCartney and Christopher D. Manning (2008). Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*, pp. 521–528. Manchester, UK.

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.

Christopher D. Manning and Hinrich Schütze (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

Mitchell P. Marcus, Beatrice Santorini and Mary A. Marcinkiewicz (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Fernando Pereira, Kiril Ribarov and Jan Hajič (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-05)*, pp. 523–530. Vancouver, BC.

Paola Merlo and Gabriele Musillo (2008). Semantic parsing for high-precision semantic role labelling. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL-2008)*, pp. 1–8. Manchester,UK.

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, Heidi Fox, Lance A. Ramshaw and Ralph M. Weischedel (2000). A novel use of statistical parsing to extract information from text. In *Proceedings of the Meeting of the North American Association for Computational Linguistics*, pp. 226–233. Seattle, Washington.

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 and Jun'ichi Tsujii (2002). Maximum entropy estimation for feature forests. In *Proc. of Human Language Technology Conf.(HLT-2002)*. San Diego, CA.

Yusuke Miyao and Jun'ichi Tsujii (2005). Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pp. 83–90. Ann Arbor, Michigan.

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

Stephen H. Muggleton, ed. (1992). *Inductive Logic Programming*. Academic Press, New York, NY.

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

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin and Dragomir Radev (2004). A smorgasbord of features for statistical machine translation. In *Proceedings of Human Language Technology Conference / North American Association for Computational Linguistics Annual Meeting (HLT-NAACL-2004)*, pp. 161–168. Boston, MA.

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

Marius Paşca (2009). Outclassing Wikipedia in open-domain information extraction: Weakly-supervised acquisition of attributes over conceptual hierarchies. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pp. 639–647. Athens, Greece.

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

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 and Ivan A. Sag (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Hoifung Poon and Pedro Domingos (2009). Unsupervised semantic parsing. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP-09)*, pp. 1–10. Suntec,Singapore.

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-04)*. 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.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin and Daniel Jurafsky (2005a). Support vector learning for semantic argument classification. *Machine Learning*, 60:11–39.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin and Daniel Jurafsky (2005b). Semantic role labeling using different syntactic views. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pp. 581–588. Ann Arbor, MI.

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 (1999). Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–176.

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.

Frank Rosenblatt (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Erik Tjong Kim Sang (2002). Memory-based shallow parsing. *Journal of Machine Learning Research*, 2:559–594.

Fei Sha and Fernando Pereira (2003). Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology Conference / North American Association for Computational Linguistics Annual Meeting (HLT-NAACL-2003)*, pp. 134–141. Edmonton, Canada.

Libin Shen and Aravind K. Joshi (2005). Incremental ltag parsing. In *Proceedings of the Human Language Technology Conference and Conference on Empirical*

Methods in Natural Language Processing (HLT/EMNLP-05), pp. 811–818. Vancouver, Canada.

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

Noah A. Smith and Jason Eisner (2005). Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL'05*, pp. 354–362. Ann Arbor, Michigan.

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

Mark Steedman and Jason Baldridge (to appear). Combinatory categorial grammar. In Robert Borsley and Kersti Borjars, eds., *Constraint-based Approaches to Grammar: Alternatives to Transformational Syntax*. Oxford: Blackwell.

Michale Strube and Simone Paolo Ponzetto (2007). Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI-2007)*. Vancouver, BC. To Appear.

Charles Sutton and Andrew McCallum (2005). Joint parsing and semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005) Shared Task on Semantic Role Labeling*. Ann Arbor, MI.

Takaaki Tanaka, Francis Bond, Timothy Baldwin, Sanae Fujita and Chikara Hashimoto (2007). Word sense disambiguation incorporating lexical and structural semantic information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural*

Language Learning (EMNLP/CoNLL-07), pp. 477–485. Prague, Czech Republic.

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*, pp. 466–477. Freiburg, Germany.

Richard Thomason, ed. (1974). *Formal Philosophy: Selected Papers of Richard Montague, edited and with an introduction by Richard Thomason*. Yale University Press.

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.

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

Kristina Toutanova, Aria Haghighi and Christopher D. Manning (2005). Joint learning improves semantic role labeling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pp. 589–596. Ann Arbor, MI.

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.

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.

Yuk Wah Wong (2007). *Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques*. Ph.D. thesis, Department of Computer Sciences, University of Texas, Austin, TX. Also appears as Artificial Intelligence Laboratory Technical Report AI07-343.

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

Yuk Wah Wong and Raymond J. Mooney (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, pp. 960–967. Prague, Czech Republic.

William A. Woods (1970). Transition network grammars for natural language analysis. *Communications of the Association for Computing Machinery*, 13:591–606.

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.

Szu-ting Yi (2007). *Robust Semantic Role Labeling Using Parsing Variations and Semantic Classes*. Ph.D. thesis, University of Pennsylvania.

Szu-ting Yi and Martha Palmer (2005). The integration of syntactic parsing and semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005) Shared Task on Semantic Role Labeling*. Ann Arbor, MI.

Dmitry Zelenko, Chinatsu Aone and Anthony Richardella (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.

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 21st 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-07)*, pp. 678–687. Prague, Czech Republic.

Luke S. Zettlemoyer and Michael Collins (2009). Learning context-dependent mappings from sentences to logical form. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pp. 976–984. Suntec,Singapore.

Vita

Ruifang Ge was born in Mengcheng, Anhui Province in China in 1976. In 1994, she went to Tsinghua University in China to study Computer Science and Technology where she obtained a Bachelor degree and later a Master degree. Ruifang is now pursuing her Doctorate degree in Computer Sciences at the University of Texas at Austin. Her research interests include natural language processing and machine learning. She will be joining the Facebook team in Palo Alto, CA after graduation.

Permanent address: Department of Computer Sciences
Taylor Hall 2.124
University of Texas at Austin
Austin, TX 78712
USA

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.