Also appears as Technical Report UT-AI-TR-06-327, Artificial Intelligence Lab, University of Texas at Austin, February 2006.

Learning Semantic Parsers Using Statistical Syntactic Parsing Techniques

Ruifang Ge
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
grf@cs.utexas.edu

Doctoral Dissertation Proposal

Supervising Professor: Raymond J. Mooney

Abstract

Most recent work on semantic analysis of natural language has focused on "shallow" semantics such as word-sense disambiguation and semantic role labeling. Our work addresses a more ambitious task we call semantic parsing where natural language sentences are mapped to complete formal meaning representations. We present our system SCISSOR based on a statistical parser that generates a semantically-augmented parse tree (SAPT), in which each internal 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. Training the system requires sentences annotated with augmented parse trees. We evaluate the system in two domains, a natural-language database interface and an interpreter for coaching instructions in robotic soccer. We present experimental results demonstrating that SCISSOR produces more accurate semantic representations than several previous approaches on long sentences.

In the future, we intend to pursue several directions in developing more accurate semantic parsing algorithms and automating the annotation process. This work will involve exploring alternative tree representations for better generalization in parsing. We also plan to apply discriminative reranking methods to semantic parsing, which allows exploring arbitrary, potentially correlated features not usable by the baseline learner. We also propose to design a method for automating the SAPT-generation process to alleviate the extra annotation work currently required for training SCISSOR. Finally, we will investigate the impact of different statistical syntactic parsers on semantic parsing using the automated SAPT-generation process.

Contents

1 Introduction

Spectacular developments have been made in natural language processing in recent years, mainly focusing on surface level tasks such as part-of-speech tagging, chunking and syntactic parsing (Manning & Schütze, 1999), while fewer efforts have been devoted to analyze meanings deep in natural language sentences. Researches on shallow semantic analysis tasks, such as word-sense disambiguation (Ide & Jeanéronis, 1998) and semantic role labeling (Carreras & Màrquez, 2005, 2004; Gildea & Palmer, 2002), only partially tackle this problem by identifying the meanings of target words or finding phrases that fill in the semantic roles of a single predicate in a sentence. Our work, however, addresses a more ambitious task we call semantic parsing, where natural language sentences are mapped to complete formal meaning representations. Semantic parsing is useful in many practical tasks such as speech understanding (Price, 1990; Zue & Glass, 2000), question answering (Lev et al., 2004), and advice taking (Kuhlmann et al., 2004).

Prior learning approaches to semantic parsing (Price, 1990; Zue & Glass, 2000) have mainly concentrated on relatively simple domains, such as the Air Travel Information Services (ATIS) domain, where the semantic parsing task can be simplified to fill a single non-recursive semantic frame. Other learning approaches have been aimed at generating meaning representations in the structure of multiple, recursive semantic frames, however, these approaches are mainly based on deterministic parsing (Zelle & Mooney, 1996; Kate, Wong, & Mooney, 2005), which lacks the robustness of statistical parsing, especially on long sentences.

In this proposal, we present an approach based on a statistical parser that generates a *semantically augmented parse tree* (SAPT), in which each internal node includes both a syntactic and semantic label. We augment Collins (1997) parsing model 2 to incorporate a semantic label on each internal node. By integrating syntactic and semantic interpretation into a single statistical model and finding the globally most likely parse, an accurate combined syntactic/semantic analysis can be obtained. Once a SAPT is generated, an additional step is required to translate it into a final formal *meaning representation* (MR) with a nested structure. Our approach is implemented in a system called SCISSOR (Semantic Composition that Integrates Syntax and Semantics to get Optimal Representations). Training the system requires sentences annotated with both gold-standard SAPTs and MRs. We present initial experimental results on real-world data sets demonstrating that SCISSOR produces more accurate semantic representations than several previous approaches on long sentences.

In future work, we will pursue several directions in developing more accurate semantic parsing algorithms and automating the labeling process:

1. Exploring alternative tree representations of SAPTs for optimal parsing accuracy;
2. Applying discriminative reranking methods to semantic parsing that would lead to improved parsing performance;
3. Designing a method for automating the SAPT-generation process to alleviate the extra annotation work currently required by training SCISSOR. We plan to use statistical syntactic parsers to generate syntactic parses automatically;
4. Investigating the impact of different statistical syntactic parsers on semantic parsing.

The remainder of this proposal is organized as follows. Section 2 briefly reviews prior work on statistical syntactic parsing, as well as several learning approaches on semantic parsing and their application domains. Section 3 presents our semantic parsing algorithm SCISSOR and some initial experimental results. We discuss proposed directions for future work in Section 4.

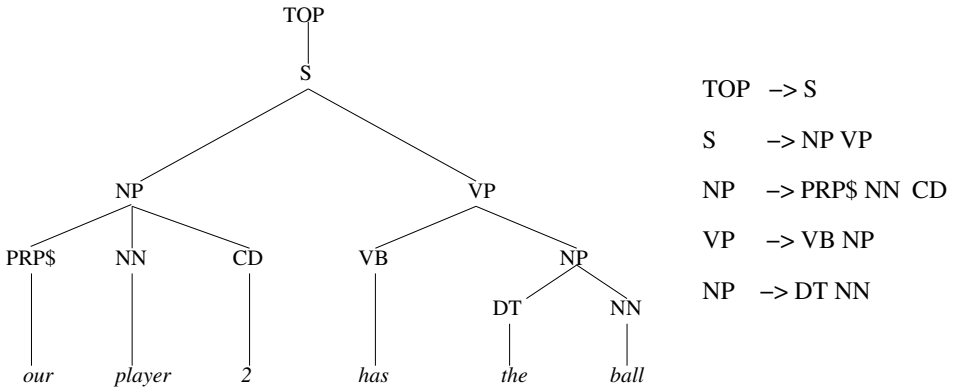


Figure 1: A syntactic parse tree and the list of rewrite rules associated.

2 Background and Related Work

2.1 Statistical Syntactic Parsing and Collins (1997) Parsing Models

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 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 preposition 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:

$$G = (N, \Sigma, S, R) \tag{1}$$

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 $LHS \rightarrow RHS$, where $LHS \in N$ and RHS is a sequence of terminals and non-terminals; each rule has an associated probability where the probabilities of all rules 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 $LHS_i \rightarrow RHS_i, 1 \leq i \leq n$.

$$\mathcal{P}(T, S) = \prod_{i=1}^{n} \mathcal{P}(LHS_i \rightarrow RHS_i) \tag{2}$$

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

A well-known drawback with PCFGs is their lack of lexicalization – the probabilities of rules are independent of words involved. For example, in the Penn Treebank (Marcus, Santorini, & Marcinkiewicz, 1993), the probabilities of the rule VP→V NP with different verbs *take* (32.1%) and *come* (1.1%) are different (Manning & 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). 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*).

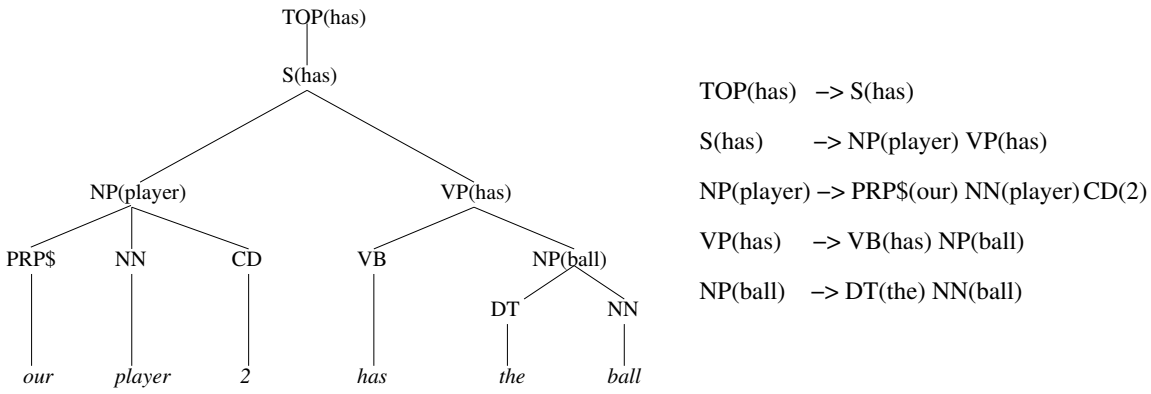


Figure 2: A lexicalized parse tree and the list of lexicalized rewrite rules associated.

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).

In the following part of this section, we introduce Collins' (1997) parser, one of the best lexicalized statistical parsers up-to-date. 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 a 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 improved performance at each time. 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 the Wh-movement where subcategorization 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 significantly improvement over CM2 (Collins, 1997). Another reason to use CM2 is that the statistics on moved complements required for training CM3 is 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 would be estimated as:

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

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 decision is associated with a probability. While history-based models have many advantages, it can be awkward to incorporate discriminative features, because the choice of features are 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, which incorporate arbitrary discriminative features, to directly choose the best parse tree. In practice, however, the algorithms become infeasible when a large exponential number of candidate trees exist, because there is no feasible way to find the best tree effciently when arbitrary features are included. Dynamic programming techniques cannot apply in this situation, and the algorithms need to enumerate over all parse trees to find the best tree.

Reranking approaches (Collins, 2000; Charniak & Johnson, 2005) address this problem with the advantages 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.

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 3: The perceptron training algorithm.

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:

$$score(x, y) = \Phi(x, y) \cdot \bar{W} \tag{3}$$

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). It also includes 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 effecient and effective in practical problems despite its age and simplicity.

The perceptron training algorithm is shown in Figure 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 are set to 0 initially. 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 are kept unchanged. Collins (2002a) gives theoretical analysis of 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 parameter vectors used in testing is the average of each parameter vector generated during the training process.

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. The parse trees in Figure 4 taken from Collins and Koo (2005) are used for illustration. The head of the rule VP→VBD NP NP SBAR in Figure 4(a) is VBD.

1. Rules. These are the counts of unique context-free rules in a syntactic parse. For example, the tree in

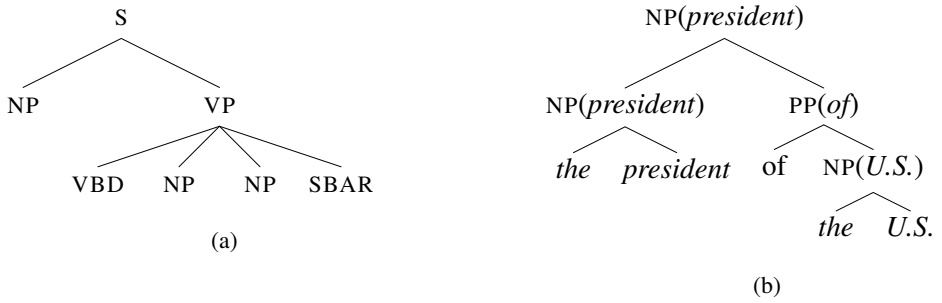


Figure 4: The syntactic parse trees for illustrating the reranking features.

Figure 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 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 4(a) has a feature f(VP→ NP NP SBAR, S)=1, where S is the non-terminal above the rule VP→ 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 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 4(a) has a feature f(VP→ NP NP SBAR, S→ 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 4(a) has a feature f(NP NP, *right*, VP, S→ 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 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 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 preposition 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 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.

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, Gildea, Khudanpur, Sarkar, Yamada, Fraser, Kumar, Shen, Smith, Eng, Jain, Jin, & Radev, 2004) and speech recognition (Collins, Koehn, & Kucerova, 2005).

2.3 Application Domains

Semantic parsing has mainly focused on three application domains. The earliest application domain is the Air Travel Information Services (ATIS) domain (Price, 1990) mainly used in speech community. 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:

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

The second domain is the GEOQUERY domain. GEOQUERY is a logical query language for a small database of U.S. geography containing about 800 facts. This domain was originally chosen to test corpus-based semantic parsing due to the availability of a hand-built natural-language interface, GEOBASE, supplied with Turbo Prolog 2.0 (Borland International, 1988). The GEOQUERY language consists of Prolog queries augmented with several meta-predicates (Zelle & Mooney, 1996). Below is a sample query with its English gloss:

```
answer(A,count(B,(city(B),loc(B,C),const(C,countryid(usa))),A))
```
How many cities are there in the US?

Kate et al. (2005) later developed a functional, variable-free version of the query language. Two corpora are developed for this domain, where the smaller corpus is the subset of the larger one. The smaller corpus contains 250 queries for the GEOQUERY database, collected by asking undergraduate students in a language class. Queries were then manually translated into the logical form (Zelle & Mooney, 1996). The average length of the NL sentence and MR are 6.76 and 6.20 tokens, respectively. There are altogether 159 unique NL tokens in the corpus. The second corpus also includes an additional 630 queries collected from an undergraduate AI class and the users of a database query interface on the web. The average length of the NL sentence and MR are 7.48 and 6.47 tokens, respectively. There are altogether 270 unique NL tokens in the corpus.

The third domain 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 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. As described in Chen et al. (2003), its grammar consists of 37 non-terminal symbols and 133 productions. Below is a sample rule with its English gloss:

```
((bpos (penalty-area our))
 (do (player-except our {4})
     (pos (half our))))
```
If the ball is in our penalty area, all our players except player 4 should stay in our half.

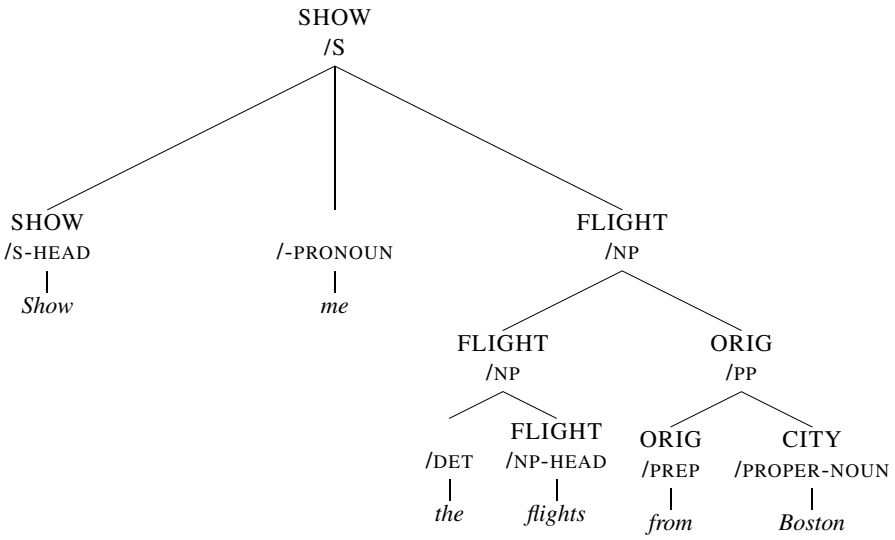


Figure 5: A sample semantically-augmented parse tree in the ATIS domain, similar to that in Miller et al. (1996).

The corpus developed for this domain (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. The average length of the NL sentence and MR are 22.52 and 13.42 tokens, respectively. There are altogether 337 unique NL tokens in the corpus.

2.4 Semantic Parsing Approaches

2.4.1 Syntax-Driven Approaches

An elegant, natural and well-studied approach in semantic parsing is the syntax-driven approach, a major approach in computational semantics (Blackburn & Bos, 2005). In this approach, each node on the tree has both a syntactic label and the semantics associated with this node; the analysis of semantics is driven by the structure of a parse tree where the meaning of the parent is built from its children in the tree. The semantics of complex linguistic constructs such as coordination, and relative clauses can be treated elegantly. Figure 5 shows a sample semantically-augmented parse tree similar to that in Miller, Stallard, Bobrow, and Schwartz (1996). The semantic label on each node is the main predicate of its subtree. Nodes having no corresponding concept in the application domain have empty semantic labels.

Miller et al. (1994, 1996, 2000) extend statistical syntactic parsing models to incorporate a semantic label on each node. Miller et al. (1994, 1996) propose a hidden understanding model for parsing, while Miller et al. (2000) utilizes a head-driven model similar to that in Collins (1997) (Section 2.1). The training data includes the annotated parse trees, thus parameter estimation is just frequency counting.

Zettlemoyer and Collins (2005) utilizes a combinatory categorial grammar (CCG) (Steedman, 2000) in semantic parsing, where each syntactic label specifies its valency and the directionality of its syntactic arguments, and each subtree has an MR attached to its root. Figure 6 gives a simple CCG parse tree taken from Zettlemoyer and Collins (2005), where the syntactic label (S\NP)/NP of the word *border* specifies that it takes an NP to the right (\NP), and then an NP to the left (/NP) to form an S (sentence) constituent, while

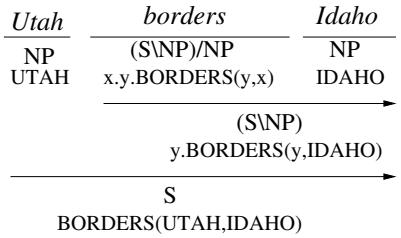


Figure 6: A CCG parse tree taken from Zettlemoyer and Collins (2005), where the meaning representation of the sentence is BORDERS(UTAH,IDAHO).

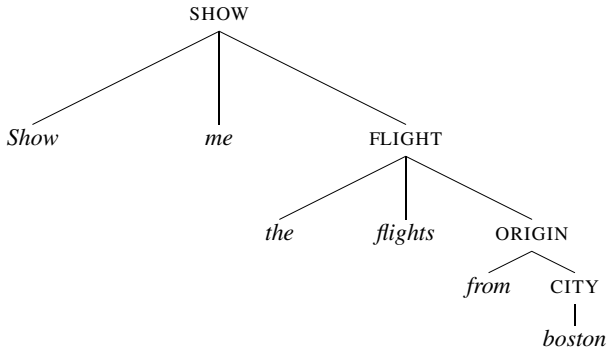


Figure 7: A sample semantic parse tree in the ATIS domain.

the semantic label $\lambda x.\lambda y.$ BORDERS(y, x) specifies that it takes the MR associated with the first and second NPs as its second and first arguments, respectively. Training on a set of sentences only labeled with their meaning representations, it induces a maximum entropy model to represent the distribution of syntactic and semantic structures for generating correct meaning representations. It requires prior knowledge of syntax for representing the semantics. A set of rules are carefully designed to specify possible syntactic categories for each type of predicates in the semantics, in which each rule is in the form of an input trigger, and an output category. Below is a sample rule for constants:

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

Using this rule, the constant IDAHO in the MR of Figure 6 would match the input pattern, and output a category NP:IDAHO. This category would then be associated with all words as their candidate labels. During training, spurious labels such as the category NP:IDAHO attached to the word *borders* would be pruned.

2.4.2 Semantic-Driven Approaches

While syntax-driven approaches have their advantages of computating semantics in its simplicity and powerfulness, syntactic parse trees can be more sophisticated than needed for meaning composition. For example, in Figure 5, since *the* conveys no meaning, the noun phrase *the flights* actually has the same meaning as its child *flights*, there is no meaning composition going on in this phrase. A more straightforward way for semantic parsing is to be driven directly by semantic representations.

One important work of these approaches is semantic grammar (Jurafsky & Martin, 2000), where the non-terminals on the tree are purely semantic labels, and each rewriting rule corresponding to some meaning composition (see Figure 7). A semantic parse tree can be seen as generated using a semantic grammar. Kate et al. (2005) employs such an approach in a system called SILT. The training data in SILT only includes the pairs of NL sentences and their MRs, while semantic parse trees are hidden, thus the key work in the algorithm is to induce a semantic grammar. The examples are first labeled as positive and negative according to the appearances of a semantic concept in the MRs, and then the rules for that concept are learned from these examples. For example, for the concept FLIGHT in Figure 7, if the pattern *the flights* ORIGIN is seen very often in the positive examples and very rare in the negative ones, then it is very likely that FLIGHT→*the flights* ORIGIN is a rule. The rules are learned in a bottom-up manner: the rules for simple concepts are learned first, and the rules for the concepts including these simple ones are learned subsequently. In the example in ATIS domain above, the rules for CITY would be learned before the rules for ORIGIN.

Wong (2005) introduce another method to induce a semantic grammar (the synchronous context-free grammar) by using machine translation techniques in a system called WASP. 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.

Hard matching of semantic rules in the above approaches can limit the coverage of a semantic grammar in testing. To address this problem, Kate (2005) introduces a system called KRISP, using a *soft* semantic grammar based on string-kernel-based classification. For each production in the meaning representation language, a classifier is trained using string similarity as the kernel to acquire the probability that this production occurs in a substring of an input sentence. Meaning representations of sentences are obtained by finding the most probable semantic parse using these probabilities.

Instead of using semantic grammars to build the semantic parse tree explicitly, He and Young (2005) treats 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 semantic labels starting from the pre-terminal of the word, and ending at the root node. As an example, the word *from* in Figure 7 would have a label ⟨ORIGIN,FLIGHT,SHOW⟩. A HMM model is trained on the sentences only labeled with their MRs, and the labels are hidden. A drawback of the encoding here 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* would have different labels (FLIGHT.ORIGIN and TRAIN.ORIG) in the system, the information from the previous example will not be helpful.

One of the earliest learning approaches of semantic parsing system (Zelle & Mooney, 1996) called CHILL learns a shift-reduce parser with control rules using induction logic programming (ILP) (Muggleton, 1992). Control rules are used to decide the parsing actions, such as when a semantic concept should be introduced, and when these concepts should be combined to build the MR. No actions are labeled for learning the control rules in the training data, therefore an ILP approach is used to induce these control rules. The learning algorithm requires a lexicon with the words paired with their possible semantic concepts as its input, which can be acquired using lexicon learning methods (Thompson & Mooney, 2003). CHILL is later extended by Tang and Mooney (2001) in a system called COCKTAIL, which uses multiple clause constructors to obtain more expressive power than a single clause constructor.

For a class of simple semantic parsing tasks, which is popular in speech understanding, such as booking the flights (ATIS) or querying the weather, the semantic parsing task can be simplified to fill a single semantic frame, where the structure among these fillers become less important. For example, the seman-

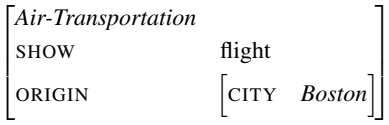


Figure 8: The semantic frame associated with the example in Figure 7.

tic frame associated with the example used above is shown in Figure 8. One of the earliest such systems, CHANEL (Kuhn & 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. For example, a match of the pattern *from city_name* would indicate the presence of an attribute ORIGIN in the query. The decision trees are learned from a set of NL sentences and their SQL queries.

3 Completed Research

We present an approach based on a statistical parser that generates a *semantically augmented parse tree* (SAPT), in which each internal node includes both a syntactic and semantic label. We augment Collins' (1997) parsing model 2 to incorporate a semantic label on each internal node to capture the predicate-argument structure of a sentence. By integrating syntactic and semantic interpretation into a single statistical model and finding the globally most likely parse, an accurate combined syntactic/semantic analysis can be obtained. Once a SAPT is generated, an additional step is required to translate it into a final formal *meaning representation* (MR) in a *meaning representation language* (MRL).

Our approach is implemented in a system called SCISSOR (Semantic Composition that Integrates Syntax and Semantics to get Optimal Representations). Training the system requires sentences annotated with both gold-standard SAPTs and MRs. We present experimental results on corpora for both geography-database querying and Robocup coaching demonstrating that SCISSOR significantly outperforms other systems (Wong (2005), Kate (2005), Tang and Mooney (2001)) on long sentences, where syntax is crucial for meaning composition.

3.1 Semantic Parsing Framework

This section describes our basic framework for semantic parsing, which is based on a fairly standard approach to compositional semantics (Jurafsky & Martin, 2000). First, a statistical parser is used to construct a SAPT that captures the semantic interpretation of individual words and the basic predicate-argument structure of the sentence. Next, a recursive procedure is used to compositionally construct an MR for each node in the SAPT from the semantic label of the node and the MRs of its children. Syntactic structure provides information of how the parts should be composed. Ambiguities arise in both syntactic structure and the semantic interpretation of words and phrases. By integrating syntax and semantics in a single statistical parser that produces a SAPT, we can use both semantic information to resolve syntactic ambiguities and syntactic information to resolve semantic ambiguities.

In a SAPT, each internal node in the parse tree is annotated with a semantic label. Figure 9 shows the SAPT for a simple sentence in the CLANG domain. Note that the semantic labels on the internal nodes capture the predicate-argument structure of the sentence, such as in the rule (omitting syntactic labels) *bowner* $\rightarrow$ *player bowner*, where the predicate *bowner* (ball owner) takes a *player* as its argument.

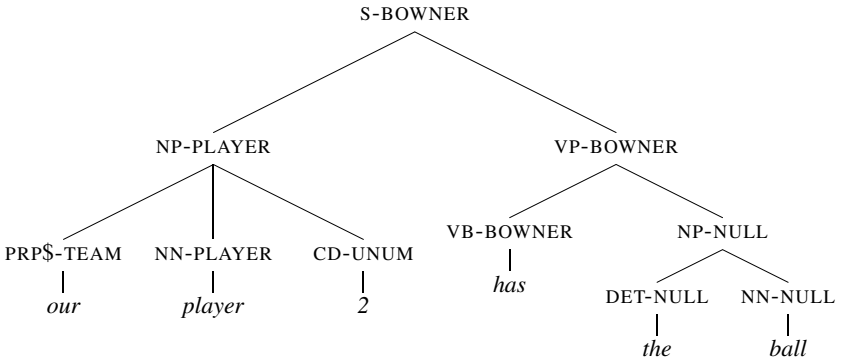


Figure 9: A SAPT for a simple CLANG sentence.

Function:BUILDMR(N, K)
Input: The root node N of a SAPT;
predicate-argument knowledge, K, for the MRL.
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 3.1
$C_{h_{MR}}$ = BUILDMR(C_h, K)
for each other child C_i where $i \neq h$
$C_{i_{MR}}$ = BUILDMR(C_i, K)
COMPOSEMR($C_{h_{MR}}, C_{i_{MR}}, K$) // see Section 3.1
$N_{MR} = C_{h_{MR}}$

Figure 10: Computing an MR from a SAPT.

The semantic labels which are shown after dashes are *concepts* in the domain. Some *type concepts* do not take arguments, like *team* and *unum* (uniform number). Some concepts, which we refer to as *predicates*, take an ordered list of arguments, like *player* and *bowner*. The predicate-argument knowledge, K, specifies, for each predicate, the semantic constraints on its arguments. Constraints are specified in terms of the concepts that can fill each argument, such as *player*(*team*, *unum*) and *bowner*(*player*). A special semantic label $null$ is used for nodes that do not correspond to any concept in the domain.

Figure 10 shows the basic algorithm for building an MR from an SAPT. Figure 11 illustrates the construction of the MR for the SAPT in Figure 9. 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. In the example, node N3 is determined to be the semantic head of the sentence, since its semantic label, *bowner*, matches N8's semantic label. Next, the MR of the semantic head is constructed recursively. The semantic head of N3 is clearly N1. Since N1 is a part-of-speech (POS) node, its semantic label directly determines its MR, which becomes *bowner*(_). Once the MR for the head is constructed, the MR of all other (non-head) children are computed recursively, and COMPOSEMR 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, N2 has a *null* label, the MR of N3 also becomes *bowner*(_). When computing the MR for N7, N4 is determined to be the head with the MR: *player*(_,_). COMPOSEMR then assigns N5's MR to fill the *team* argument

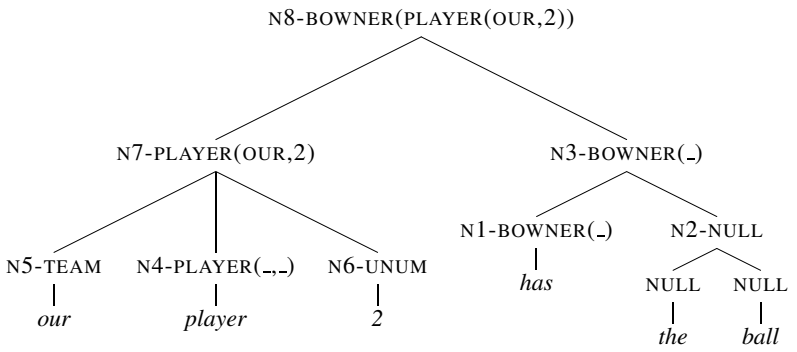


Figure 11: MRs constructed for each SAPT Node.

and N6's MR to fill the *unum* argument to construct N7's complete MR: *player*(*our*, *2*). This MR in turn is composed with the MR for N3 to yield the final MR for the sentence: *bowner*(*player*(*our*,*2*)).

For MRLs, such as CLANG, whose syntax does not strictly follow a nested set of predicates and arguments, some final minor syntactic adjustment of the final MR may be needed. In the example, the final MR is *(bowner (player our {2}))*. In the following discussion, we ignore the difference between these two.

3.2 Corpus Annotation

This section discusses how sentences for training SCISSOR were manually annotated with SAPTs. Sentences were parsed by Collins (1997) parser (Bikel, 2004) (trained on sections 02-21 of the WSJ Penn Treebank) to generate an initial syntactic parse tree. The trees were then manually corrected and each node was augmented with a semantic label using the following procedure.

First, semantic labels for individual words, called *semantic tags*, are added to the POS nodes in the tree. The tag *null* is added to words that have no corresponding concept in the application domain. For concepts conveyed by phrases, like "has the ball" for *bowner* in the previous example, only one word is labeled with the concept, where the syntactic head word (Collins, 1997) is preferred; the other words in the phrase provide context for determining the semantic label of the word during parsing.

After that, labels are added to the remaining nodes in a bottom-up manner. For each node, one of its children is chosen as the semantic head, from which it inherits the label. The semantic head is chosen to be the child whose semantics take other children's semantics as arguments in the MR. For example, in Figure 9, the root node inherits the semantic label from its semantic head VP-BOWNER, which takes NP-PLAYER as an argument in the MR. This step was done mostly automatically, but required some manual corrections to account for unusual cases.

In order for COMPOSEMR to be able to construct the MR for a node, the argument constraints for its semantic head must identify a unique concept to fill each argument. However, some predicates take multiple arguments of the same type, such as *point.num*(*num*,*num*), which is a subclass of point that represents a field coordinate in CLANG. In this case, extra nodes are inserted in the tree with new type concepts that are unique for each argument. An example is shown in Figure 12 in which the additional type concepts *num1* and *num2* are introduced. Again, during parsing, context will be used to determine the correct type for a given word.

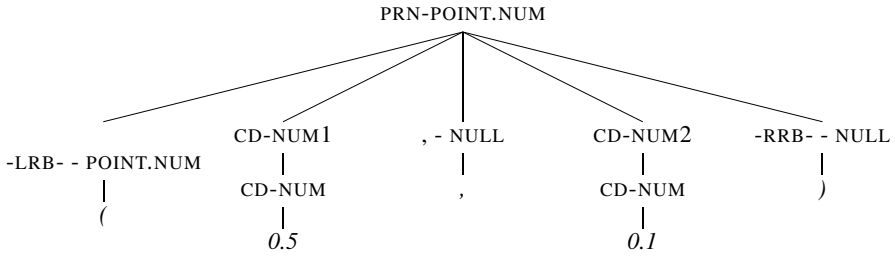


Figure 12: Adding new types to disambiguate arguments.

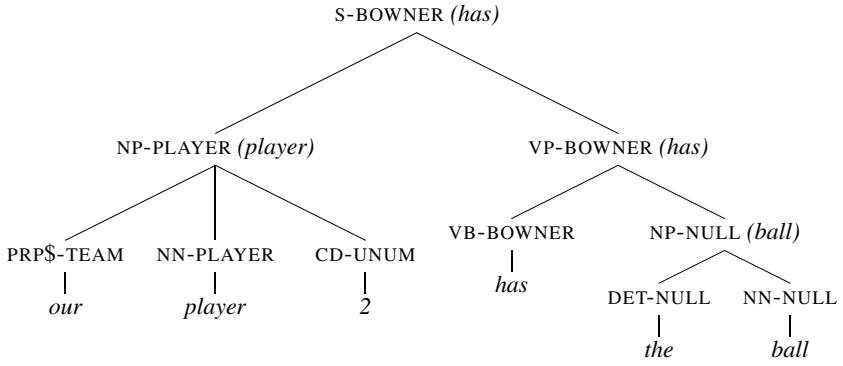


Figure 13: A lexicalized SAPT.

3.3 Integrated Parsing Model

3.3.1 Integrating Semantics into the Model

SCISSOR introduces a parsing model for generating SAPTs for sentences, by extending Collins (1997) parsing model 2 (see Section 2.1 for detailed description) to incorporate the generation of semantic labels in SAPTs. We focus on the extensions in the following discussion.

In SCISSOR, the representation of each non-terminal in a SAPT is extended to include a semantic label, together with a syntactic label; each non-terminal is lexicalized with both a head word, part-of-speech tag and also a semantic tag passed up from its head child. Figure 13 shows a lexicalized SAPT (omitting the POS tag and semantic tag lexicalized to each non-terminal).

The generation of the right-hand-side (RHS) of a rule in SAPTs is decomposed in the same way as in Collins (1997): first generating the head of the RHS, then generating the modifiers conditioned on the head. The difference is that each child has the new representation augmented with semantics. To model a predicate's argument preference for generating semantic labels of modifiers, each head is predicted with a semantic subcategorization frame, in addition to a syntactic subcategorization frame. For example, the predicate *player* in CLANG requires a *team* and a *unum* as its arguments. The implementation of semantic subcat frames is similar to syntactic subcat frames, which are multi-sets specifying the semantic-label requirements of a head to its left or right modifiers. The generation of modifiers is modified to be constrained on both syntactic subcats and semantic subcats.

We introduce the new meanings of notations for describing the extended model. The subscript syn refers to the syntactic part, and sem refers to the semantic part. We redefine X and x to include semantics:

BACK-OFFLEVEL	$\mathcal{P}_h(H\|...)$	$\mathcal{P}_{LC}(LC\|...)$	$\mathcal{P}_{L1}(L_i\|...)$	$\mathcal{P}_{L2}(lt_i\|...)$	$\mathcal{P}_{L3}(lw_i\|...)$
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 1: Conditioning variables for each back-off level in semantic parameters.

each non-terminal X is a pair of a syntactic label X_{syn} and a semantic label X_{sem} ($X = \langle X_{syn}, X_{sem}\rangle$); x is a triple of a word w, a POS tag t_{syn}, and a semantic tag t_{sem} ($x = \langle w, t_{syn}, t_{sem}\rangle$). Left and right subcat frames LC and RC are redefined as: $\langle LC_{syn}, LC_{sem}\rangle$ and $\langle RC_{syn}, RC_{sem}\rangle$. All other notations are the same as in Section 2.1.

The probability of a rule $LHS \rightarrow RHS$ in the form (P is the parent, H is the head child, and L and R are the left and right children):

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

is calculated as the product of the following probabilities in the same way as described in Section 2.1:

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 subcat 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 the 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 generating the phrase "our player 2" in Figure 13 using the rule NP-player → PRP$-[team](our) NN-player CD-[unum](2) is calculated as (omitting the distance measure):

$\mathcal{P}_h$(NN-[player]|NP-[player],player)×
$\mathcal{P}_{lc}$(⟨{},{team}⟩|NP-[player],player)×$\mathcal{P}_{rc}$(⟨{},{unum}⟩|NP-[player],player)×
$\mathcal{P}_l$(PRP$-[team](our)|NP-[player],player,⟨{},{team}⟩)×$\mathcal{P}_l$(STOP|NP-[player],player,⟨{},{}⟩)×
$\mathcal{P}_r$(CD-[unum](2)|NP-[player],player,⟨{},{unum}⟩)×$\mathcal{P}_r$(STOP|NP-[player],player,⟨{},{}⟩)

3.3.2 Smoothing

The non-terminals in the extended model are the combinations of syntactic labels and semantic labels, and thus need more smoothing. To address this problem, the parameters (probability estimations in the generation steps) are decomposed using the chain rule as follows (only the parameters for generating the left modifiers are shown here):

$$\mathcal{P}_h(H|C) = \mathcal{P}_{h_{syn}}(H_{syn}|C) \times \mathcal{P}_{h_{sem}}(H_{sem}|C,H_{syn}) \quad (4)$$

$$\mathcal{P}_{lc}(LC|C) = \mathcal{P}_{lc_{syn}}(LC_{syn}|C) \times \mathcal{P}_{lc_{sem}}(LC_{sem}|C,LC_{syn}) \quad (5)$$

$$\mathcal{P}_l(L_i(l_i)|C) = \mathcal{P}_{l_{syn}}(L_{i_{syn}}(lt_{i_{syn}},lw_i)|C) \times \mathcal{P}_{l_{sem}}(L_{i_{sem}}(lt_{i_{sem}},lw_i)|C,L_{i_{syn}}(lt_{i_{syn}})) \quad (6)$$

For brevity, C is used to represent the context on which each parameter is conditioned; and lw_i, $lt_{i_{syn}}$, and $lt_{i_{sem}}$ are the word, POS tag, and semantic tag for a non-terminal L_i. Words are generated individually in both syntactic and semantic outputs.

To further simplify the model, we make the independence assumption to condition syntactic output only on syntactic features, and semantic output only on semantic features. 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 reduced to the same parameters as in Collins (1997) (Section 2.1), and are smoothed using the method in Collins (1997). In the following part of this subsection, we discuss the techniques for smoothing the semantic parameters. Since the semantic parameters do not depend on syntactic features under the independence assumption, we can safely omit the sem subscripts in the discussion. The parameter $\mathcal{P}_l(L_i(lt_i, lw_i)|P, H, w, t, \Delta, LC)$ for generating the left modifier is further decomposed as:

$$\mathcal{P}_{l1}(L_i|P, H, w, t, \Delta, LC) \times \mathcal{P}_{l2}(lt_i|P, H, w, t, \Delta, LC, L_i) \times \mathcal{P}_{l3}(lw_i|P, H, w, t, \Delta, LC, L_i(lt_i)) \quad (7)$$

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 $\mathcal{P}_{l1}$ and $\mathcal{P}_{l2}$. This is because semantic tags are essentially more specific than semantic tags, and requires more smoothing. Table 1 shows the back-off levels for each semantic parameter. The probabilities from these back-off levels are interpolated using the techniques in Collins (1997).

3.3.3 Tagging and Parsing

The parsing model does not rely on an external POS tagger or semantic tagger, instead, it uses the following method to provide candidate tags for parsing. It classifies words into known and unknown words (all words occurring less than 3 times in the training data, and words in the test data that were not seen in training). For known words, the candidate tags are those that have been seen with this 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 its associated POS tag during training. Note that the unknown word threshold (3) is smaller than the one used by Collins (1997) since the training corpora for semantic parsing are small.

In decoding, a CKY-style parser is used to find a best SAPT that maximizes the joint probability of a sentence and a SAPT.

3.4 Implementation Details

We discuss the approaches used for handling pronoun resolution and non-compositionality in SCISSOR in this section. We utilize a straightforward method to deal with pronoun resolution. In SAPTs, each pronoun is labeled with a semantic label, and the task of pronoun resolution is to find an entity having the same semantic type as the pronoun as its referee. SCISSOR chooses the latest entity that appears before the pronoun, and also has number agreement with the pronoun. For example, in the sentence "If our player 6 has the ball, then he should shoot", "he" has the semantic label *player*, thus its referee should be a *player* that appears before it, which is the case in "our player 6".

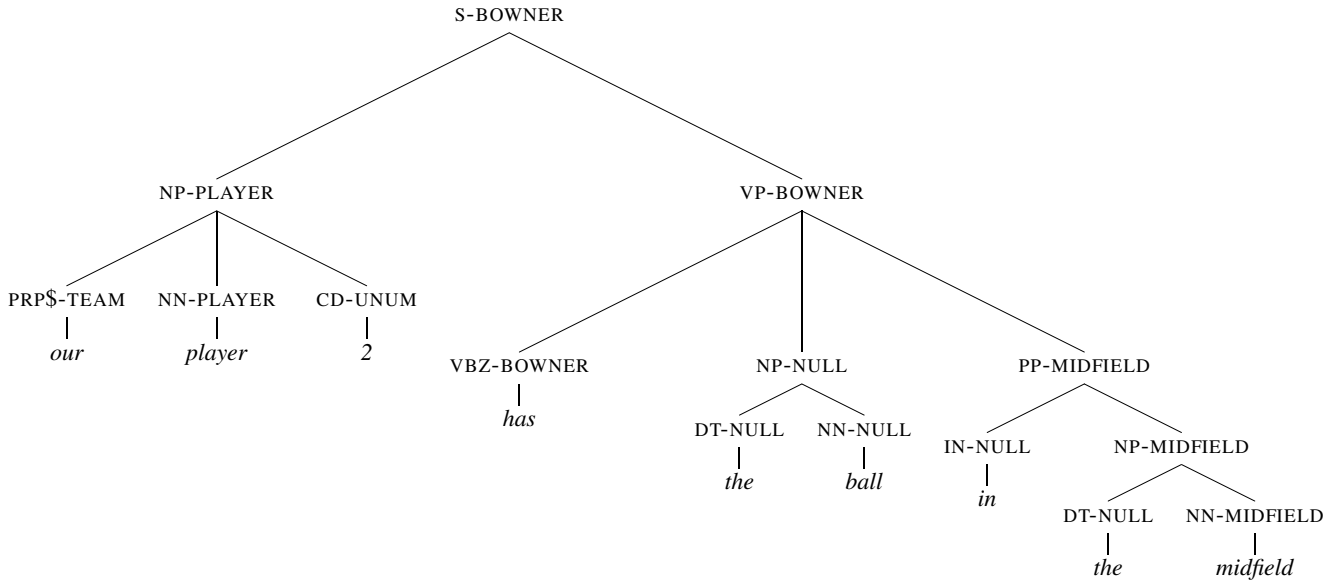


Figure 14: A simple example in CLANG illustrating non-compositionality, where the meaning representation of the sentence is (AND (BOWNER (PLAYER (OUR 2))) (BPOS (MIDFIELD))) .

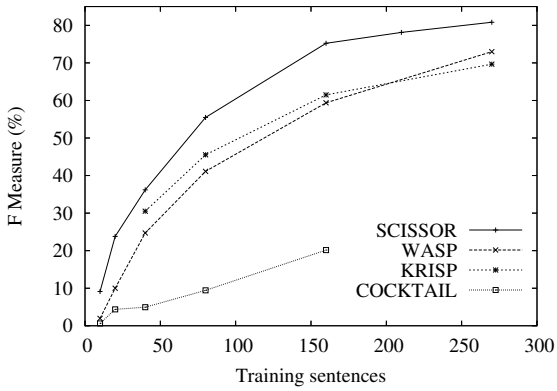
Non-compositionality occurs in the application domains, where MR structures do not correspond to the structures of syntactic parse trees. For example, in Figure 14, *midfield* in the MR is attached to the predicate *bpos* (ball position), however, in the syntactic parse tree, "in the midfield" (midfield) is attached to the verb "has" (*bowner*) instead of "the ball" as in the meaning representation. In this case, we combine the two predicates *bowner* and *bpos* to a new predicate *bowner.bpos*, which takes the arguments from both individual predicates, thus the word "has" can be augmented with the new predicate *bowner.bpos*, and the MR of the sentence can be built compositionally. Other non-compositionality cases are handled in similar ways.

3.5 Experimental Evaluation

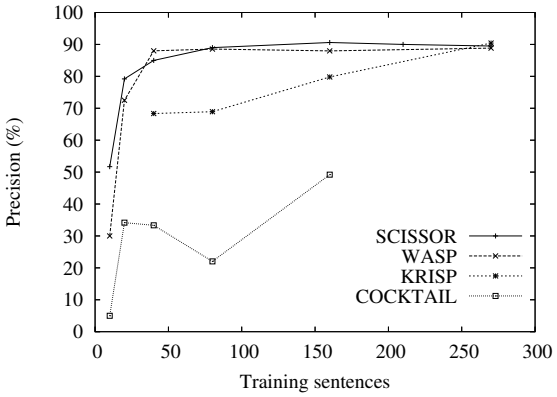
3.5.1 Methodology

We evaluated SCISSOR on three corpora introduced in Section 2.3: CLANG, the small GEOQUERY corpus, namely GEO250, and the large GEOQUERY corpus, namely GEO880. The average NL sentence lengths (in tokens) of the three copora are 22.51, 6.72, and 7.48, respectively, and the average number of possible semantic tags for each word that can represent meanings is 1.59 in CLANG, and 1.46 in GEO250. SCISSOR has not been evaluated in the ATIS domain, however, Popescu et al. (2004) showed that parsing the GEOQUERY data is harder than parsing the ATIS data.

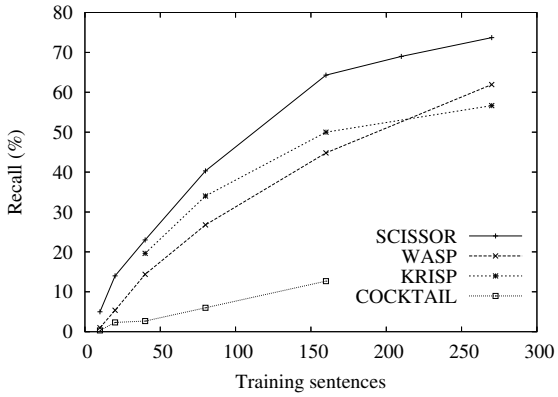
SCISSOR was evaluated using standard 10-fold cross validation. NL test sentences were first parsed to generate their SAPTs, and then their MRs were built from the trees. The unknown word threshold was 3, and the beam width was 10^4 (a parse with its probability within $\frac{1}{10^4}$ of the probability of the top parse in the same chart entry is kept). We measured the number of test sentences that produced complete MRs, and the number of these 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 GEOQUERY, an MR is correct if



(a) F-measure curves



(b) Precision curves



(c) Recall curves

Figure 15: Learning curves on CLANG.

the resulting query retrieved the same answer as the correct representation when submitted to the database. The performance of the parser was then measured in terms of *precision*, *recall*, and *F-measure*:

$$\text{Precision} = \frac{\text{Number of correct MRs}}{\text{Number of test sentences with completed MRs}} \quad (8)$$

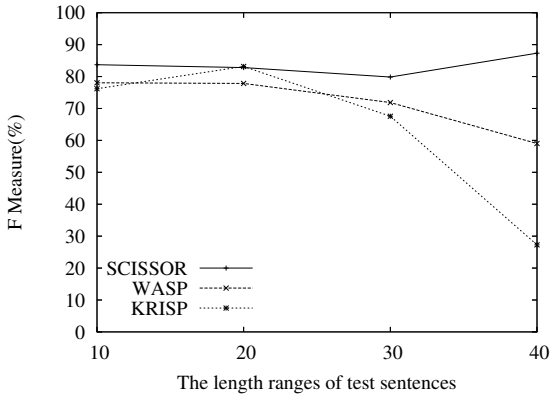
$$\text{Recall} = \frac{\text{Number of correct MRs}}{\text{Number of test sentences}} \quad (9)$$

$$\text{F-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision+Recall}} \quad (10)$$

We compared SCISSOR on all corpora to three other algorithms: WASP (Wong, 2005) , KRISP (Kate, 2005) and COCKTAIL (Tang & Mooney, 2001). WASP and KRISP are two most recently developed, unsupervised statistical semantic parsing algorithms, where WASP is based on synchronous context-free grammar, and KRISP is based on string-kernel-based classification. COCKTAIL is an unsupervised deterministic shift-reduce parser based on inductive logic programming. We also compared SCISSOR to Zettlemoyer

Length range	0-10	11-20	21-30	31-40	41-50
Count of sentences	22	98	137	38	5
Corresponding point in the figure below	*10*	*20*	*30*	*40*	–

(a) Sentence length distribution



(b) Results

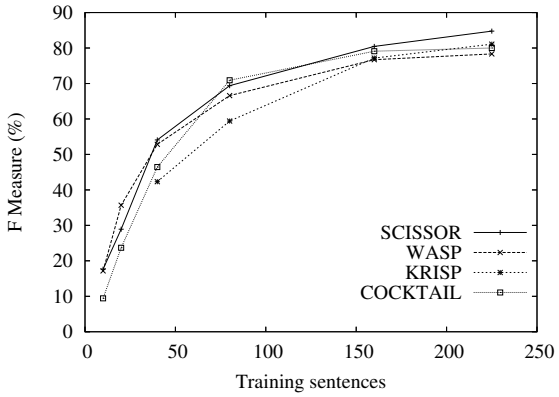
Figure 16: Results for test sentences within certain length ranges on CLANG.

and Collins (2005), with results reported on GEO880. Zettlemoyer and Collins (2005) introduced an integrated syntactic-semantic parser based on combinatory categorical grammar (CCG), which requires a set of domain-specific hand-built rules for building a CCG lexicon. All these systems do not need full syntactic-semantic annotation of the training examples. In the evaluation, all these systems except Zettlemoyer and Collins (2005) performed 10-fold cross validation using the same splits between training and test data. Since KRISP only presented plotted precision-recall (PR) curves, and did not give unique precision and recall to compare with, we used the confidence that maximizes the F-measure on the last data point of a learning curve to be the threshold for each corpus, and used the precision and recall at the threshold for each data point to generate learning curves. Zettlemoyer and Collins (2005) used a different experimental method on GEO880, where the data was divided into 600 training examples and 280 test examples, and the reported result was averaged over two runs over the data.

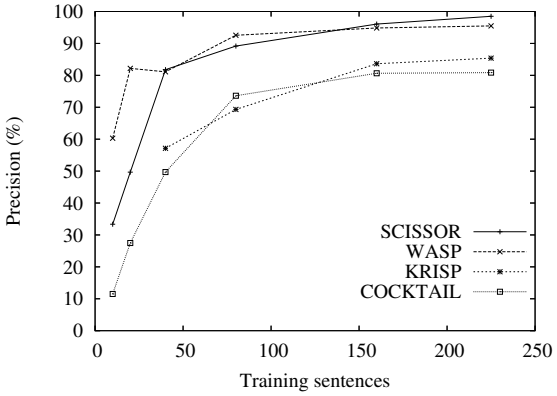
3.5.2 Results

Figure 15 shows the learning curves on CLANG. SCISSOR performs significantly better than other systems, with the recall about 12% higher than WASP, and 17% higher than KRISP. COCKTAIL shows unsatisfying results on this corpus: it can not handle training sets larger than 160 examples due to its intensive memory requirement. It also shows poor performance on other data points because of its deterministic decision-making nature – once it fails to parse a sentence in the middle, it does not backtrack. This drawback becomes especially problematic on CLANG where sentences are long.

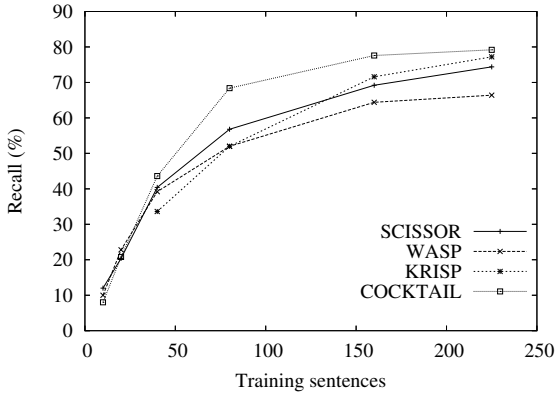
Figure 16 gives detailed look at the F-measures on sentences within different length ranges on CLANG. Figure 16(a) lists the sentence counts in different length ranges, and their corresponding points in Fig-



(a) F-measure curves



(b) Precision curves



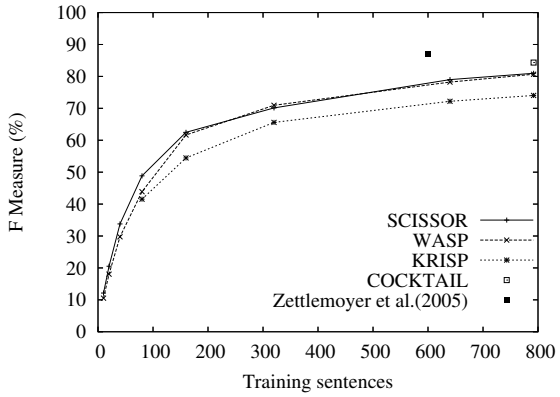
(c) Recall curves

Figure 17: Learning curves on GEO250.

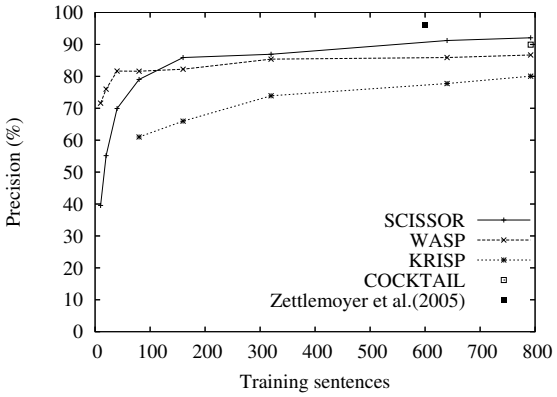
ure 16(b); the performance of sentences within length range 41-50 is not shown in Figure 16(b), because only 5 sentences fall into this category. It shows that great improvement of SCISSOR over WASP and KRISP appears when sentences are long: while WASP and KRISP degrade significantly as sentences are getting longer, SCISSOR does not show such degradation. This suggests that the meaning composition process guided by syntax employed in SCISSOR is especially useful when sentences are long.

Figure 17 gives the learning curves on GEO250. While SCISSOR shows higher F-measure on the last two data points, overall, it does not show substantial improvement as in CLANG. This is consistent with the results in CLANG, where great improvement of SCISSOR over other systems appears when sentences are long. Figure 18 shows the learning curves on GEO880. SCISSOR has lower F-measure than both Zettlemoyer and Collins (2005) and COCKTAIL. Careful quantitative analysis shows that the errors relate closely to the annotation method for augmenting semantic labels to the internal nodes of SAPTs in GEOQUERY. We will give more detail in Section 4.1, and it is also a part of our future work.

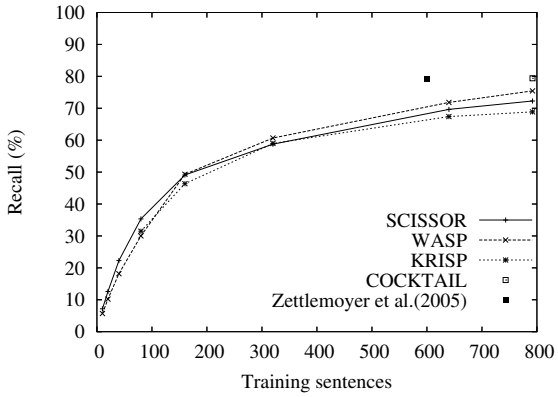
Overall, the experiments demonstrate that SCISSOR significantly outperforms other systems on long



(a) F-measure curves



(b) Precision curves



(c) Recall curves

Figure 18: Learning curves on GEO880.

sentences, where syntax is crucial for meaning composition.

4 Proposed Research

4.1 Better Generalization in Tree Representation

In Section 3.5.2, our experiments show that SCISSOR gives lower F-measure than both Zettlemoyer and Collins (2005) and COCKTAIL on GEO880. Careful analysis suggests that the low F-measure could be caused by the annotation method for adding semantic labels to the internal nodes of SAPTs (see Section 3.2).

Before discussing the annotation problem, let us first briefly introduce the meaning representation language namely FUNQL used in GEOQUERY. Each predicate in meaning representations in FUNQL is treated as a function modifying an argument list. For example, in the MR of Figure 19, the innermost expression, COUNTRYID('USA') represents a single element: the US; and the predicate LOC_2 is applied to the list,

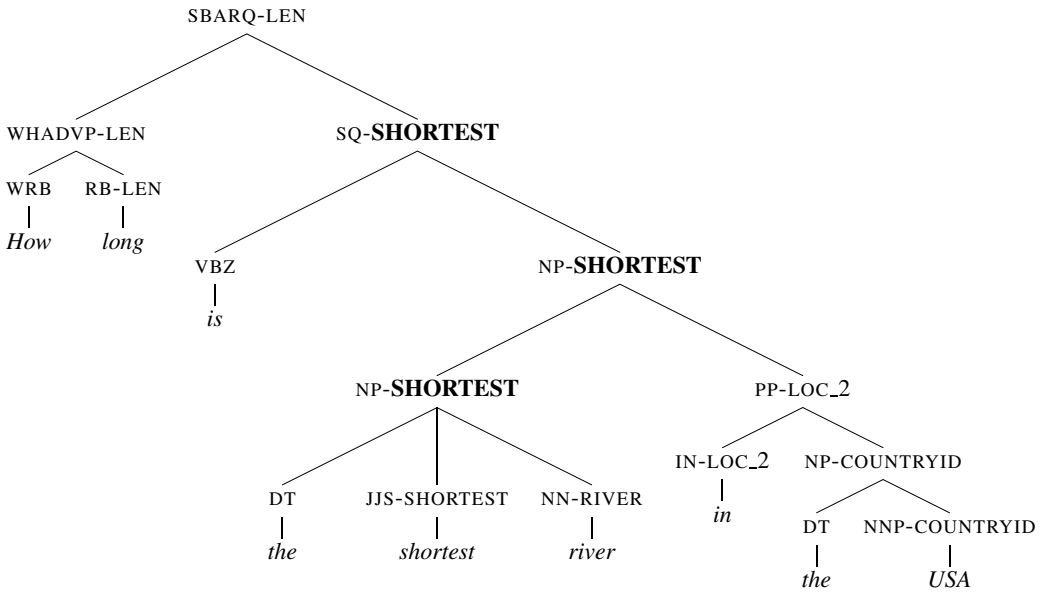


Figure 19: An example in GEOQUERY illustrating the labeling problem, where the meaning representation of the sentence is ANSWER(LEN(SHORTEST(RIVER(LOC_2(COUNTRYID('USA')))))). The annotation is produced using the method in Section 3.2.

so that the expression LOC_2(COUNTRYID('USA')) returns the list of entities in the US. The same process goes on so that the expression RIVER(LOC_2(COUNTRYID('USA'))) returns the list of rivers in the US, the expression SHORTEST(RIVER(LOC_2(COUNTRYID('USA')))) returns the shortest river in the US, and the expression LEN(SHORTEST(RIVER(LOC_2(COUNTRYID('USA'))))) returns the length of the shortest river in the US. Finally, the special predicate ANSWER returns the acquired elements, which is the length of the shortest river in the US in this example.

The annotation process described in Section 3.2 adds semantic labels to the non-leave nodes of a syntactic parse tree in a bottom-up manner. The semantic label of each node is the same as the semantic label of one child, whose semantics takes other children's semantics as the arguments in the meaning representation. Applying this process to the tree in Figure 19, the semantic label of the phrase *the shortest river* will be SHORTEST, because RIVER is the argument of SHORTEST in the meaning representation. The label SHORTEST is further passed up the tree according to the same procedure.

We can see that semantic labels added in this way do not resemble the dependency relation in the syntactic side. For example, the phrase *the shortest river* has the syntactic head *river*, however, in the semantic side, it has the same semantic label (SHORTEST) as another child *shortest* to represent a subset of RIVER. Learning from these SAPTs can potentially lead to severe sparse data problem. For example, the statistics acquired from Figure 19 only relates to a subset of RIVER namely SHORTEST, while the statistics on other subsets of RIVER, such as LONGEST, have to be learned separately. Quantitative analysis on GEO880 for the experiments in Section 3.5 demonstrates that error rates in the sentences that includes predicates like SHORTEST are significantly higher than that in the other sentences.

A simple and principled way to tackle this problem is to utilize the productions in the meaning representation language, where the left-hand-side (LHS) of a production provides generalization for the concept in the right-hand-side (RHS). These productions have been explored by the semantic parsing systems in Wong

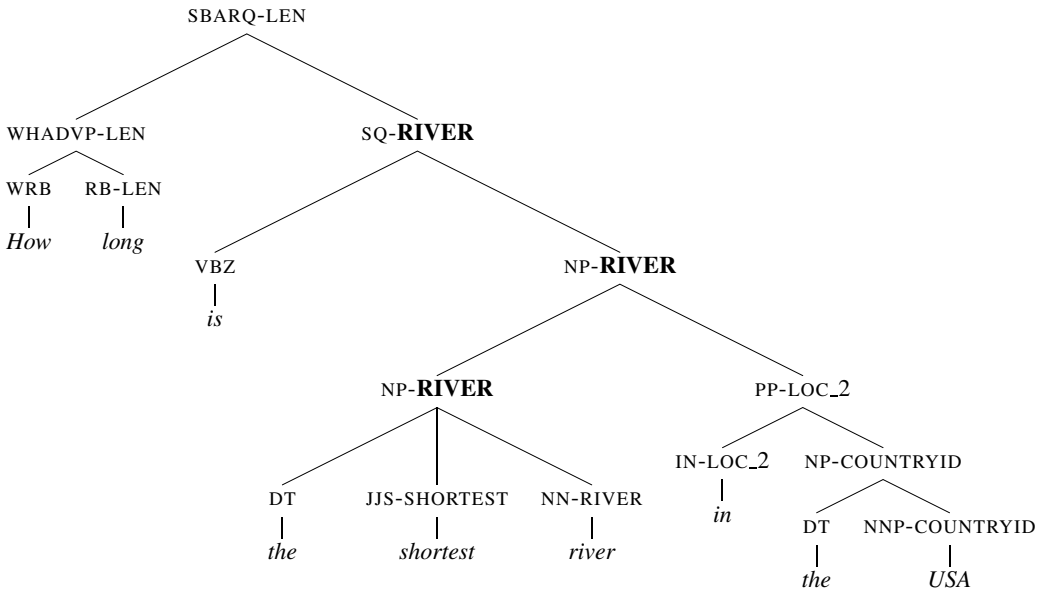


Figure 20: The new annotation of the example in Figure 19.

(2005) and Kate (2005). For example, in the production rule RIVER → SHORTEST RIVER in FUNQL, the LHS (RIVER) is the superclass of the RHS (SHORTEST RIVER). Each production rule is associated with a pattern describing its meaning representation. For example, in the production rule above, the pattern specifies that SHORTEST would enclose RIVER. In annotation, instead of labeling a node with the semantic label passed up from its child, it can be labeled with the LHS of the production involved. Thus, the phrase *the shortest river* in Figure 19 will be labeled as RIVER using the production rule RIVER → SHORTEST RIVER. The new SAPT annotated using production rules is shown in Figure 20. Note that the statistics acquired from the new SAPT relates to all rivers (RIVER).

By providing better generalization in tree labels, which still capture the predicate-argument structures of sentences, we hope that training SCISSOR on the newly-annotated GEOQUERY corpora will produce better results. The same procedure of generating meaning representations as in Section 3.1 can still be used here.

4.2 Discriminative Reranking for Semantic Parsing

In this section, we discuss ongoing work on learning to rerank the top outputs from SCISSOR, which allows exploring arbitrary, potentially correlated features not usable by the baseline learner. Detailed description for training and testing a reranking model is described in Section 2.2. Briefly, in a reranking model, semantically-augmented parse trees (SAPTs) are mapped into feature vectors, where each feature is associated with a weight representing the feature's power for predicting the correct SAPT. Training a reranking model amounts to estimating the weight associated with each feature, and in testing, the SAPT with the highest score (use Equation 3) under a weight vector is outputted.

4.2.1 Features

A central issue in reranking SAPTs is to devise a set of features for predicting correct outputs. One set of features that we can exploit is those features proven to be useful in reranking syntactic parses, such as the

BACK-OFFLEVEL	$\mathcal{P}_{L1}(L_i\|...)$
1	P,H,w,t,Δ,LC
2	P,H,t,Δ,LC
3	P,H,Δ,LC
4	P,H
5	P

Table 2: Extended back-off levels of the semantic parameter $\mathcal{P}_{L1}(L_i|...)$, using the same notation as in Section 3.3.

features introduced by Collins (2000), which are described in detail in Section 2.2. Specially, the score of a parse under a base model is included as a feature to take advantage of the baseline model. Besides using syntactic features for predicting the correctness of the syntactic part, we could extend the model to include a set of semantic features based on the correspondence between syntax and semantics – for each syntactic feature type, we introduce a similar semantic feature type. For example, the feature type *Semantic rules* is introduced corresponding to the syntactic feature type *Rules*. These are the counts of unique semantic context-free rules in a SAPT, where non-terminals are semantic labels (the empty semantic label NULL is not included). For example, the tree in Figure 13 would have the feature f(PLAYER$\rightarrow$ TEAM PLAYER UNUM)=1. For brevity, other semantic features are not listed here.

Another set of features that we can exploit is the features used in semantic role labeling (SRL). Given a sentence and a target word, SRL identifies all constituents that fill semantic roles of the target word, such as agent, patient or instrument, together with its adjuncts, such as location or manner. State-of-the-art SRL systems (Carreras & Màrquez, 2004, 2005) generally explore the following types of features: features capturing the internal structure of candidate arguments, and features extracted from the arguments' context; features describing properties of a target predicate, and features generated from the predicate's context; and also features modeling the distance between a predicate and its argument. These features should be valuable in semantic parsing for predicting predicate-argument relations.

4.2.2 Preliminary Experimental Evaluation

Methodology. We have evaluated reranking models using different feature sets adapted from (Collins, 2000) on CLANG (300 examples) and Geo250 (250 examples), adopting standard 10-fold cross validation. First, the base parsing model SCISSOR was used to generate N candidate parses (SAPTs) for training and test examples, where N was 50. After that, a reranking model was trained on the training examples using the averaged perceptron algorithm described in Section 2.2. The correct parse of each sentence required by training is a parse that lead to the correct meaning representation, and it is the parse with the highest syntactic F-measure score among these parses. If the correct parse does not exist, then the training example is discarded. In testing, the averaged weight vector was used to choose the best parse from the candidate parses. In all these experiments, the number of iterations T over the training examples was 10, and the feature-count cut-off was 0. We have experimented with other cut-offs (2 and 4), and they did not show significant improvements.

SCISSOR failed to generate enough candidate parses using the beam width and back-off levels introduced in Section 3 because of training on the small datasets. To acquire sufficient candidate SAPTs, we increased the beam width from 10^4 to 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 (8 on average), and thus it is harder to get

TRIAL	CLANG			GEO250		
	P	R	F	P	R	F
SCISSOR	**89.51**	73.76	80.88	**98.52**	74.43	84.80
SCISSOR+	86.94	**78.19**	**82.33**	95.50	**77.20**	**85.38**

Table 3: The performance of the base system SCISSOR+ compared with SCISSOR, where P refers to precision, R refers to recall, and F refers to F-measure.

TRIAL	CLANG			GEO250		
	P	R	F	P	R	F
SCISSOR+	86.94	78.19	82.33	95.50	77.20	85.38
Oracle score	-	85.58	-	-	81.60	-
sem	89.55	80.54	**84.81**	96.50	77.20	**85.78**
syn	87.31	78.52	82.68	95.07	77.20	85.21
sem+syn	88.81	79.87	84.10	95.52	76.8	85.14

Table 4: Reranking results on SCISSOR+ outputs using different feature sets.

enough candidates using a small beam width. We also extended the back-off levels of the parameters for generating modifiers' semantic labels (see Table 2, only the parameter for the left side is shown), so that SCISSOR would allow the exploration of more parses.

Results. Table 3 gives the results of the base learner SCISSOR using different parameters: SCISSOR uses the beam width and back-off levels in Section 3.3, and SCISSOR+ uses the new beam width and back-off levels. SCISSOR+ shows better recall, and worse precision than SCISSOR on both corpus. Since SCISSOR+ does not constrain the generation of modifiers' semantic labels on semantic subcat frames, and allow broader search, the result is reasonable.

Table 4 compares the results of reranking models using different feature sets adapted from (Collins, 2000). In all these experiments, the score of a SAPT in SCISSOR+ is included as a feature. The oracle score is the upper-bound a reranking model can achieve, where an oracle tells which parse is the best for each example. *sem* is the reranking model using only the semantic features, *syn* using only the syntactic features, and *sem+syn* using both the syntactic and semantic features. Only using the semantic features, *sem* achieves the best performance, with 2.48% absolute F-measure improvement (14.0% relative error reduction) in CLANG and 0.4% absolute F-measure improvement (2.7% relative error reduction) in GEO250. Note that *sem+syn* performs worse than *sem*. Though we do not want to over-interpret the small difference, the result may suggest that when training on the small datasets, the model using both syntactic and semantic features starts over-fitting. We have also experimented with introducing features combining both syntax and semantics, and they also failed to show improvement.

Overall, our results show that discriminative reranking can improve upon the baseline system SCISSOR in semantic parsing. Future work includes a further investigation of features derived from SAPTs, such as the features used in SRL. We also plan to investigate the reasons behind the modest improvement on GEOQUERY compared to CLANG.

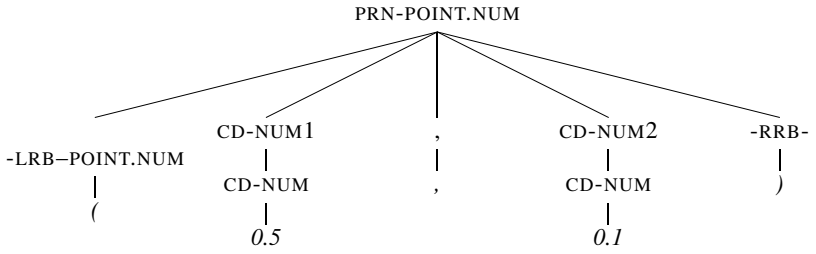


Figure 21: A SAPT in CLANG, where the meaning representation of the sentence is (POINT.NUM 0.5 0.1).

4.3 Automating the Semantically-Augmented Parse Tree Generation

Training SCISSOR requires a set of training examples augmented with semantically-augmented parse trees (SAPTs). The annotation work required by generating SAPTs restricts both the size of available training data in one domain and its application to new domains. In this section, we propose to automate the SAPT-generation process for training corpus with sentences paired with their meaning representations.

SAPTs are composed of syntactic parse trees and augmented semantic labels. Since syntactic parse trees with satisfying accuracies can be generated automatically using statistical syntactic parsers (Section 4.3.1), the key issue of the SAPT generation is to add semantic labels to these trees for composing correct meaning representations. The semantic labels on the non-leave nodes of SAPTs can be essentially classified into two types. One type of semantic labels is passed up from one of their children using a compositional-semantics process (Section 3.2), such as the root node's semantic label POINT.NUM in Figure 21. The other type of semantic labels is semantic-role labels specially introduced when a predicate can take multiple arguments of the same type. For example, in Figure 21, NUM1 and NUM2 are introduced to specify the unique arguments of the predicate POINT.NUM. This kind of labels can be introduced by exploiting the correct meaning representations for sentences.

The remaining issue for generating SAPTs is to add correct semantic labels to the words. We propose to induce a maximum entropy model similar to the one in Zettlemoyer and Collins (2005) to represent the distribution of semantic labels of words for generating correct meaning representations. No extra human annotation is required in this process. The model is trained on a set of sentences paired with their meaning representations, where each sentence is also augmented with a syntactic parse tree generated using the method in Section 4.3.1.

This model is different with the one in Zettlemoyer and Collins (2005), in which it only learns the distribution of semantic labels for training sentences, while relying on available statistical syntactic parsers to learn the distribution of syntactic structures. This is in contrast with Zettlemoyer and Collins (2005), where it learns both syntactic and semantic distributions for generating SAPTs. It relies on a set of carefully-designed rules to specify possible syntactic categories (its valency and the directionality of its arguments) for predicates with different numbers of arguments in the semantics. These rules should be sufficient for parsing the training corpus (Section 2.4.1).

The section is organized as follows. We first discuss using statistical parsing techniques to automatically acquire syntactic parse trees in Section 4.3.1. We then talk about obtaining candidate semantic labels for words in Section 4.3.2. Section 4.3.3 introduces a maximum entropy model, which discriminates among candidate label sequences of words for a sequence, which can generate the correct meaning representation. Finally, in Section 4.3.4 and 4.3.5, we discuss the methods to obtain better initial parameters for estimating a maximum entropy model on incomplete data.

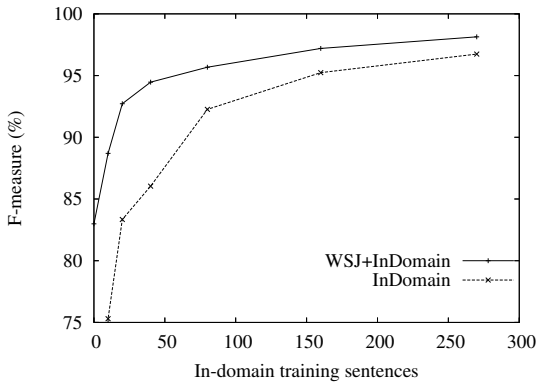


Figure 22: F-measure learning curves of syntactic parsing on CLANG.

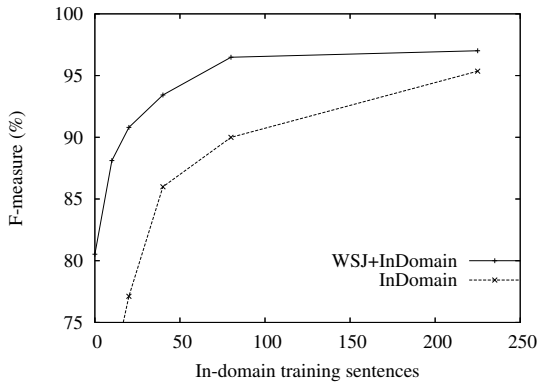


Figure 23: F-measure learning curves of syntactic parsing on GEO250.

4.3.1 Using Statistical Syntactic Parsers to Obtain Syntactic Parse Trees

Automatically generated syntactic parse trees have been successfully used in many natural language processing tasks such as semantic role labeling (Carreras & Màrquez, 2005). To take advantage of the rapid development of syntactic parsing techniques, we propose to use statistical parsers to obtain syntactic parse trees needed for generating a SAPT. These high performance parsers include parsers which output Penn Treebank (Marcus et al., 1993) style parse trees, such as Collins' (1997) parser and Charniak's (2000) parser. They also include a combinatory categorical grammar (CCG) parser (Hockenmaier & Steedman, 2002) where non-terminals in the trees encode subcategorization information. Most recently, Charniak and Johnson (2005) reported a new highest F measure, 91.02% on sentences of length less than 100, achieved so far on parsing the Penn Treebank using a reranking approach.

The most widely used corpus by statistic parsers is the Penn Treebank, a large corpus of syntactic parse trees. However, because of corpus variation, applying a syntactic parser trained only on the Penn Treebank directly to a semantic parsing corpus taken from different data distribution would critically degrade the performance of the parser (Gildea, 2001). To acquire statistics inherent in the application domain, we can train the parser either only on the in-domain parse trees or on the treebank, together with a small amount of in-domain parse trees. The benefit of the second approach is that the statistics acquired in the treebank can reduce the generalization error on unseen test examples.

We experimented with training Collins' parser (Bikel, 2004) using a few in-domain (CLANG and GEOQUERY parse trees, either together with the Penn WSJ Treebank (WSJ+InDomain) or not (InDomain), while only testing on the in-domain data. Figure 22 and 23 shows the F measure learning curves of the parser in both domains. As is clearly shown in the figures, including WSJ (WSJ+InDomain) in training significantly outperforms not using WSJ (InDomain) in both domains. For the experiments using WSJ in training, we can see that the performance of the parser degrades substantially when zero in-domain training sentence is provided, only at around 80% in the GEOQUERY domain. This is much worse than the result reported in Bikel (2004) testing on the WSJ corpus – around 90% F measure for sentences of length less than 40 words. We can also see that the performance improves quickly when adding only a small amount of in-domain parse trees– the F measure in both domains arise to above 90% when only 20 in-domain sentences are used, and around 95% when 80 are used.

Syntactic parsing errors could be corrected using a post-processing procedure as introduced in Popescu

et al. (2004). In some semantic parsing task, it is possible to acquire the domain-specific semantic constraint information from resources such as a database or a lexicon, and this information can be used to correct the syntactic errors. For example (Popescu et al., 2004), in the ATIS domain, the preposition *on* is constrained to only take the pair *flight* and *day* as its arguments, but not a *city* and a *day*. So if the preposition phrase *on Monday* (day) is incorrectly attached to *Chicago* (city) in the sentence *When are the flights from Chicago on Monday*, the error could be fixed by attaching it to *the flights*. Note that syntactic parsing errors do not necessarily hurt the performance of semantic parsing. First, what is the correct syntactic parse tree of a sentence is arguable – the correct syntactic parse tree in one linguist's view could be wrong from other linguists's view. Second, some errors, such as tagging a proper noun as a normal noun, are irrelevant to the compositional-semantics procedure used to build the meaning of a sentence, thus would not hurt the performance of semantic parsing.

4.3.2 Obtaining Candidate Semantic Labels for Words

In order to augment a sentence with a sequence of semantic labels for generating the correct meaning representation, each word in the sentence would require a set of candidate semantic labels to start with. Simply allowing each predicate in the meaning representation for each word would result in an exponential output space. To keep the search space manageable, certain techniques like beam search can be used to prune the search space during searching. Another option is to reduce the number of candidate labels for each word even before searching.

Co-occurrence measures, such as mutual information (Church & Hanks, 1990), can be used to find candidate labels for a word (Manning & Schütze, 1999). Using these measures, the predicates with the highest co-occurrence scores with the word would be chosen as its candidate labels. In addition, many existing learning systems which utilize more sophisticated techniques for finding associated predicates for a word can also be used. One of these systems is WOLFIE (Thompson & Mooney, 2003), which has been used in a semantic parser called CHILL in parsing the GEOQUERY corpus. Using a greedy search method based on a co-occurrence measure, WOLFIE acquires a semantic lexicon from a corpus of sentences paired with their meaning representations. The predicates associated with a word in the lexicon can be used as candidate labels. Other systems include machine translation systems such as the one introduced by Brown et al. (1990). In these systems, a sentence and meaning representation pair would be treated as a pair of sentences in a source and target language, respectively, and the words in the sentence would be aligned with the predicates in the meaning representation. The predicates associated with a word in the top alignments could be treated as the candidate labels for the word.

The systems introduced above are capable of augmenting sentences with semantic label sequences by themselves, thus a natural question is why we want to introduce another system for the same task. The answer lies in the definition of *correctness* for semantic label sequences in semantic parsing. We define a label sequence to be correct only if the word meanings in the sequence, combined with the sentence structure provided by the syntactic parse tree, can lead to the correct meaning representation of the sentence using a compositional-semantics procedure. Hence the systems above achieve surface level correctness, while the method proposed aimed to achieve deep level correctness for computing the semantics.

4.3.3 A Maximum Entropy Model

In the problem of labeling words in a sentence with semantic labels using its syntactic parse tree, it is ideal to select a model that can incorporate arbitrary, potentially overlapping features over the input sentence, such as dependencies among semantic labels of words. Maximum entropy models have such advantages and have

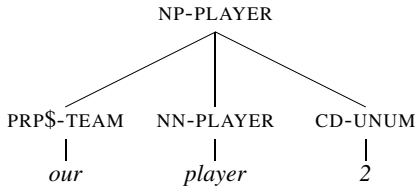


Figure 24: A simple example in CLANG illustraing the features used in a maximum entropy model.

been successfully applied to a variety of natural language tasks such as part-of-speech tagging (Ratnaparkhi, 1996; Lafferty et al., 2001), parsing (Ratnaparkhi, 1999; Clark & Curran, 2003; Abney, 1997; Johnson et al., 1999), information extraction (Borthwick et al., 1998; McCallum et al., 2000), machine translation (Berger et al., 1996; Och & Ney, 2002) , and semantic role labeling (Carreras & Màrquez, 2004). More concretely, we propose to utilize a conditional maximum entropy model similar to those used in (Riezler et al., 2002; Zettlemoyer & Collins, 2005; Clark & Curran, 2004) capable of learning on incomplete training data. The training data for this task is incomplete because a fully-supervised training would require the semantic label of each word;however, only meaning representations of sentences are annotated.

Let us first introduce the terminologies and notations we will use in describing such a model. For brevity, we will refer to *input* (S) for a sentence and its syntactic parse tree, *label sequence* (L) for the sequence of semantic labels of words in a sentence. A meaning representation can have multiple label sequences leading to it, by following the computational-semantics procedure described in Section 3.1. For training examples, a label sequence can only lead to a unique meaning representation (M), because slot ambiguities existing in the procedure can be solved using the gold standard meaning representation. We say a label sequence is *consistent* (Riezler et al., 2002) with a meaning representation when word meanings in the label sequence, combined with the syntactic parse tree in the input, can lead to the meaning representation. M^* stands for gold-standard meaning representations.

In a conditional maximum entropy model for the incomplete training data problem, the conditional probability of a label sequence L given an input S is defined as:

$$Pr(L|S;\bar{\theta}) = \frac{\exp(\bar{\theta} \cdot \bar{f}(S,L))}{\sum_{L' \in \triangle(S)} \exp(\bar{\theta} \cdot \bar{f}(S,L'))} \tag{11}$$

where $\bar{f}(S,L)$ is a function that maps the pair (S,L) into a feature vector in $\mathbb{R}^n$, parameter $\bar{\theta}$ is a vector of weights associated with each feature, $\bar{\theta} \cdot \bar{f}(S,L)$ is the inner-product of the two vectors, and $\triangle(S)$ is the set of label sequences that S can take (all possible word meaning assignments to an input sentence). Features contain information on how likely a label sequence can be used to generate the correct MR. One type of such feature used in Zettlemoyer and Collins (2005) is lexical features representing the semantic labels a word can take. Specially, the number of times a word, its semantic label and syntactic label co-occur in a tree. For example, the tree in Figure 24 has the lexical feature f(*our*,TEAM, PRP$)=1. Another type of such features could be features modeling predicate-argument structures in the semantics, which form the counterpart of dependency features in the syntax (Clark & Curran, 2004). For example, the tree in Figure 24 has the predicate-argument feature f(PLAYER, UNUM)=1.

The conditional probability of a meaning representation M given an input S is then defined as the sum of the conditional probabilities of the set of label sequences consistent with it, which we denote with $\triangle(S,M)$:

$$Pr(M|S;\bar{\theta}) = \sum_{L \in \triangle(S,M)} Pr(L|S;\bar{\theta}) \tag{12}$$

Given a set of partially-labeled training examples $\{(S_1, M_1^*), (S_2, M_2^*), ..., (S_m, M_m^*)\}$, training a conditional maximum entropy model on incomplete data involves finding a parameter $\bar{\theta}^*$ that maximizes the product of the conditional likelihood of the partially labeled training set:

$$\begin{aligned}
\bar{\theta}^* &= \arg\max_{\bar{\theta}} \prod_{j=1}^{m} Pr(M_j^*|S_j;\bar{\theta}) \\
&= \arg\max_{\bar{\theta}} \sum_{j=1}^{m} log Pr(M_j^*|S_j;\bar{\theta}) \\
&= \arg\max_{\bar{\theta}} \sum_{j=1}^{m} log \sum_{L_j \in \triangle(S_j,M_j^*)} Pr(L_j|S_j;\bar{\theta}) \\
&= \arg\max_{\bar{\theta}} \sum_{j=1}^{m} log \sum_{L_j \in \triangle(S_j,M_j^*)} \frac{\exp(\bar{\theta} \cdot \bar{f}(S_j, L_j))}{\sum_{L_j \in \triangle(S_j)} \exp(\bar{\theta} \cdot \bar{f}(S_j, L_j))}
\end{aligned} \tag{13}$$

A variety of parameter estimation methods can be used to find such a parameter $\bar{\theta}$ that optimizes the above objective function, including generalized iterative scaling (Darroch & Ratchliff, 1972) and improved iterative scaling (Della Pietra et al., 1997). These methods also include gradient-based methods, such as gradient ascent, conjugate gradient (Fletcher & Reeves, 1964), and a quasi-Newton method called limited-memory quasi-Newton (L-BFGS) (Nocedal & Wright, 1999). The experiments by Malouf (2002) show that gradient-based methods, with the exception of gradient ascent, generally converge faster than iterative methods, and L-BFGS performs the best among the gradient-based methods.

Gradient-based methods require the calculation of the gradient of the objective function in Equation 13, which we call $L(\bar{\theta})$:

$$L_{\bar{\theta}} = \sum_{j=1}^{m} log \sum_{L_j \in \triangle(S_j,M_j^*)} \frac{\exp(\bar{\theta} \cdot \bar{f}(S_j, L_j))}{\sum_{L_j \in \triangle(S_j)} \exp(\bar{\theta} \cdot \bar{f}(S_j, L_j))} \tag{14}$$

the gradient of the function with respect to θ_i is calculated as:

$$\begin{aligned}
\frac{\partial L_{\bar{\theta}}}{\partial \theta_i} &= \sum_{j=1}^{m} \sum_{L_j \in \triangle(S_j,M_j^*)} \frac{\exp(\bar{\theta} \cdot \bar{f}(S_j, L_j)) * f_i(S_j, L_j)}{\sum_{L_j \in \triangle(S_j,M_j^*)} \exp(\bar{\theta} \cdot \bar{f}(S_j, L_j))} \\
&\quad - \sum_{j=1}^{m} \sum_{L_j \in \triangle(S_j)} \frac{\exp(\bar{\theta} \cdot \bar{f}(S_j, L_j)) * f_i(S_j, L_j)}{\sum_{L_j \in \triangle(S_j)} \exp(\bar{\theta} \cdot \bar{f}(S_j, L_j))} \\
&= \sum_{j=1}^{m} \sum_{L_j \in \triangle(S_j,M_j^*)} Pr(L_j|S_j, M_j^*;\bar{\theta}) * f_i(S_j, L_j) \\
&\quad - \sum_{j=1}^{m} \sum_{L_j \in \triangle(S_j)} Pr(L_j|S_j;\bar{\theta}) * f_i(S_j, L_j)
\end{aligned} \tag{15}$$

The algorithm finds a $\bar{\theta}$ that optimizes the objective function when the gradient is 0. Note that unlike the fully-supervised case, $L(\bar{\theta})$ is not a concave function with respect to $\bar{\theta}$, and it is only *locally* maximized. Good initial parameters would be very important for obtain an optimization value that is closer to the global maxima, and we propose to use supervision to obtain these parameters (see Section 4.3.4 and 4.3.5). When such supervision is not available, all θ_i should be set to zero initially to assume as little as possible. To prevent over-fitting, a Gaussian prior (Chen & Rosenfeld, 1999) should be used to penalize large weights.

The first term of Equation 15 is the expectation of feature f_i over all label sequences consistent with each gold standard meaning representation in the training data, and the second term is the expectation of feature f_i over all label sequences that each S can take . Calculation of these values requires summing over all label sequences that a sentence can take, and summing over all label sequences consistent with a gold standard meaning representation. Since both sets can be extremely large, it is not feasible to directly enumerate them. Fortunately, it is possible to collect these statistics using a variant of the inside-outside algorithm based on a packed chart (Geman & Johnson, 2002; Miyao & Tsujii, 2002; Clark & Curran, 2004) since the features are generated using a tree structure. In a packed chart, chart entries that are equivalent for producing features are packed together.

4.3.4 Improving Parameter Estimation Using Semi-Supervised Learning

The objective function (see Equation 14) for estimating a maximum entropy model is not concave with respect to $\bar{\theta}$, so the model estimation is sensitive to initial parameters. To obtain good initial parameters, we propose to use semi-supervised learning (Seeger, 2000), where a large amount of partially-labeled or unlabeled data, together with a small amount of fully-labeled data, is used jointly to build a good model. The justification for using partially-labeled or unlabeled data to obtain a better model can be found in (Nigam et al., 2000). In the task of learning semantic label sequences for sentences, the fully-labeled data are sentences with words labeled with meanings, and the partial-labeled data are sentences only labeled with meaning representations while the label sequences are unknown. More specifically, we propose to use a method similar to the one used by Nigam et al. (2000) where the model will be first trained on the data labeled with label sequences to obtain good initial parameters, and then it is retrained on both examples labeled with label sequences, and examples only labeled with meaning representations using the initial parameters to obtain a better model.

In a semi-supervised learning setting, a few sentences with words annotated with semantic labels are added to the training data, hence the training data include both a set of examples labeled with meaning representations $\{(S_1, M_1^*), (S_2, M_2^*)$,...,$(S_m, M_m^*)\}$ and a set of examples labeled with both meaning representations and label sequences $\{(S_{m+1}, M_{m+1}^*, L_{m+1}^*), (S_{m+2}, M_{m+2}^*, L_{m+2}^*), ..., (S_{m+n}, M_{m+n}^*,$ $L_{m+n}^*)\}$. The same notation utilized in Section 4.3.3 is used here, where S is an input sentence and its syntactic parse tree, M is the meaning representation of S, and L is the label sequence of S. M^* and L^* stands for gold-standard meaning representations and label sequences, respectively.

As a first step, the model obtains the initial parameters $\bar{\theta}_0$ that maximize the product of the conditional probabilities of a label sequence L given an input S, trained only on the examples labeled with label sequences:

$$\bar{\theta}_0^* = \arg\max_{\bar{\theta}} \prod_{j=m+1}^{m+n} Pr(L_j^* | S_j; \bar{\theta}) \tag{16}$$

After that, the model is retrained on both the partially-labeled and fully labeled data, using the initial parameters $\bar{\theta}_0$ just learned, where training the model amounts to maximizing the product of the conditional

probability of the meaning representation M given S. The conditional probability of a meaning representation given an input S on the fully-labeled data is the same as the conditional probability of the label sequence L of S:

$$\begin{aligned}
\bar{\theta}^* &= \arg\max_{\bar{\theta}} \prod_{j=1}^{m} Pr(M_j^*|S_j;\bar{\theta}) * \prod_{j=m+1}^{m+n} Pr(M_j^*|S_j;\bar{\theta}) \\
&= \arg\max_{\bar{\theta}} \left(\prod_{j=1}^{m} \sum_{L_j \in \triangle(S_j, M_j^*)} Pr(L_j|S_j;\bar{\theta}) * \prod_{j=m+1}^{m+n} Pr(L_j^*|S_j;\bar{\theta}) \right) \\
&= \arg\max_{\bar{\theta}} \left(\sum_{j=1}^{m} \log \sum_{L_j \in \triangle(S_j, M_j^*)} Pr(L_j|S_j;\bar{\theta}) + \sum_{j=m+1}^{m+n} \log Pr(L_j^*|S_j;\bar{\theta}) \right) \qquad (17)
\end{aligned}$$

Here $\triangle(S_j, M_j^*)$ are the set of label sequences that are consistent with M_j^*. A weighting factor ranging between 0 and 1 can be added to the first term of the above equation to adjust the strength of the unlabeled data (Nigam et al., 2000).

4.3.5 Improving Parameter Initialization Using Glossaries

Meaning representation languages (MRL) are often accompanied by a manual describing the language using natural language sentences, and this information can also be used to suggest good initial parameter values to the model described in Section 4.3.3. For example, CLANG is provided with a manual (Chen et al., 2003) describing the predicates in the language. Below we give two definitions excerpted from the manual:

1. (POS REGION)
   The player should position itself in REGION.
2. (DRIBBLE REGION)
   The ball should be dribbled to REGION.

where POS and BPOS are predicates in CLANG which both require an argument with a semantic type REGION.

Predicates are often named after the words carrying their meaning, using either the whole word or the abbreviation of the word, thus the string similarity between a predicate and the tokens in its definition can be used to infer the likelihood of a token representing a predicate. For example, in the above definitions, the predicate POS is named after *position*, and DRIBBLE is named after *dribble*. The orthographical similarity score can be acquired using string similarity measures, such as string edit distance (Levenshtein, 1966).

The similarity between the definition of a predicate and the context where a token appears in the training corpus can also be seen as a good indicator of a token's meaning (Lesk, 1986); a high similarity score between a token and a predicate would suggest a high initial parameter value to the associated feature in the maximum entropy model. For example, in the training corpus, if the token *position* co-occurs very often with the words *player* and *itself* (words in POS's definition), but not other words, then the parameter associated with the feature which says that *position*'s semantic label is POS should have a high initial value. Though the token *player* appears with *position* and *itself*, it also co-occurs very often with other words like *pass*, which is not in the definition of POS, thus it is less similar to POS compared with *position*. The context of a token in the training corpus is defined as the combination of the context of each occurrence of the token, where each context of the token's single occurrence is the words around the token within some limited window size. A

variety of methods can be utilized to measure the similarity, among which, one of the most commonly used method is cosine similarity. In this method, both the definition and context are represented as vectors, where each element of the vectors is the weight of an associated token. The cosine similarity is then calculated as the cosine of the angle between the two vectors d_1 and d_2:

$$sim(d_1, d_2) = \frac{d_1 \cdot d_2}{\|d_1\|\|d_2\|}$$

where $\|d_1\|$ and $\|d_2\|$ are the lengths of the two vectors, respectively, and $d_1 \cdot d_2$ is their inner-product. The vectors will get a high similarity value if they are similar, and vice versa .

We use a simple example to illustrate this method. Assume the weight of a token is the number of times it appears in a definition or the training corpus, and the token vector associated with weight vectors is (*player, position, itself, pass*), then, using the statistics in the training data, POS's definition is represented as (1,1,1,0) (d_1), *position*'s context is (10,10,10,0) (d_2), and *player*'s context is (20,10,10,10) (d_3). Cosine similarities between the definition and the two tokens are calculated as:

$$\begin{aligned} sim(d_1, d_2) &= \frac{d_1 \cdot d_2}{\|d_1\|\|d_2\|} = \frac{1*10+1*10+1*10+0*0}{\sqrt{3}\cdot 10\sqrt{3}} = 1 \\ sim(d_1, d_3) &= \frac{d_1 \cdot d_3}{\|d_1\|\|d_3\|} = \frac{1*20+1*10+1*10+0*10}{\sqrt{3}\cdot 10\sqrt{7}} = 0.87 \end{aligned}$$

As we can see, *position* has a higher similarity value than *player*.

The syntactic relations in the definitions resemble the predicate-argument relations in the semantics. Besides using context, these syntactic relations can also be used to help infer how likely a token is to represent a predicate. For example, in the definition of the predicate POS, if we assume that the head of the definition should resemble the predicate it defines, then the word *position* should have the meaning POS. The definition can be further used to infer the tokens representing the argument. The token *position* is connected with its argument REGION through a PP-attachment using the preposition *in*. If the same syntactic relation also appears in the training corpus, such as in the sentence *Player 2 should position itself in the midfield*, then the semantic type of the word which appears in the same place as REGION in the syntactic relation (*midfield* here) is very likely to be a REGION. The hypothesis could be further reinforced if other definitions also suggest the same thing. For example, the definition of the predicate BPOS given previously provides another syntactic relation where a REGION could appear. If the word *midfield* also appears in sentences which resemble this syntactic relation, then we can be more confident about the meaning of this word. We can use utility measures combining the evidences to suggest good inital parameter values in maximum entropy models.

One problem with dictionary-based methods is that the definitions are usually pretty short and could have little overlap with the context of a token (Manning & Schütze, 1999). Lexical resources such as WordNet (Miller, 1991) and other thesauruses can be used to expand definitions as suggested by researches on word sense disambiguation (Pook & Catlett, 1988) and information retrieval (Baeza-Yates & Ribeiro-Neto, 1999).

4.4 Evaluating the Impact of Statistical Syntactic Parsers

Various natural language processing tasks utilizing syntactic parse trees have tried to evaluate the impact of using automatic syntactic parses on the accuracies of the application tasks, such as the work by Gildea and Palmer (2002) in semantic role labeling. It is also a crucial question we want to answer in our work based on

statistical syntactic parsing techniques. We have experimented with training SCISSOR on SAPTs augmented with gold-standard parses in Section 3. In the future, we plan to also train SCISSOR on SAPTs augmented with automatic syntactic parses (Section 4.3.1) to measure the impact of using gold-standard and automatic parses in the performance.

We also plan to investigate semantic parsing performances using automatic parses generated by different statistical syntactic parsers. State-of-the-art statistical parsers include Treebank parsers, such as Collins (1997) parser and Charniak (2000) parser, and combinatory categorical grammar (CCG) parsers, such as Hockenmaier and Steedman (2002) parser and Clark and Curran (2004) parser. The CCG formalism is known for its elegant treatment of linguistic phenomena such as coordination and relative clauses, where non-terminals encode syntactic subcat information. We plan to analyze the impact of using parsers based on different tree representations and parsing models to semantic parsing.

5 Conclusion

We have presented a semantic parsing approach based on a statistical parser that generates a semantically-augmented parse tree. Once a SAPT is generated, a compositional-semantics procedure is used to translate it into a final formal meaning representation with a nested structure. Preliminary experimental results on real-world data sets demonstrate that SCISSOR produces more accurate semantic representations than several previous approaches on long sentences. In the future, we plan to explore alternative tree representations for better generalization in parsing. We contemplate applying discriminative reranking methods to semantic parsing, which allows exploring arbitrary, potentially correlated features not usable by the baseline learner. We also propose to design a method for automating the SAPT-generation process to alleviate the extra annotation work currently required for training SCISSOR. Finally, we will investigate the impact of different statistical syntactic parsers on semantic parsing using the automated SAPT-generation process.

References

Abney, S. (1997). Stochastic attribute-value grammars. *Computational Linguistics*, *23*(4), 597–698.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. ACM Press, New York.

Berger, A. L., Della Pietra, S. A., & Della Pietra, V. J. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, *22*(1), 39–71.

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

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

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

Borthwick, A., Sterling, J., Agichtein, E., & Grishman, R. (1998). Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Proc. of 6th Workshop on Very Large Corpora*.

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

Carreras, X., & Màrquez, L. (2005). Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proc. of 9th Conf. on Computational Natural Language Learning (CoNLL-2005)*, pp. 152–164, Ann Arbor, MI.

Carreras, X., & Màrquez, L. (2004). Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proc. of 8th Conf. on Computational Natural Language Learning (CoNLL-2004)*, Boston, MA.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proc. of 14th Natl. Conf. on Artificial Intelligence (AAAI-97)*, pp. 598–603, Providence, RI.

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

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

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

Chen, S. F., & Rosenfeld, R. (1999). A Gaussian prior for smoothing maximum entropy model. Technical report CMU-CS-99-108, School of Computer Science, Carnegie Mellon University.

Church, K. W., & Hanks, P. W. (1990). Word association norms, mutual information and lexicography. *Computational Linguistics*, *16*(1), 22–29.

Clark, S., & Curran, J. R. (2003). Log-linear models for wide-coverage CCG parsing. In *Proc. of the 2003 Conf. on Empirical Methods in Natural Language Processing (EMNLP-03)*, pp. 97–105, Sapporo, Japan.

Clark, S., & Curran, J. R. (2004). Parsing the WSJ using CCG and log-linear models. In *Proc. of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pp. 104–111, Barcelona, Spain.

Collins, M. (2000). Discriminative reranking for natural language parsing. In *Proc. of 17th Intl. Conf. on Machine Learning (ICML-2000)*, pp. 175–182, Stanford, CA.

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

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

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

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

Collins, M. (2004). Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In Harry Bunt, J. C., & Satta, G. (Eds.), *New Developments in Parsing Technology*. Kluwer.

Collins, M., Koehn, P., & Kucerova, I. (2005). Discriminative syntactic language modeling for speech recognition. In *Proc. of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pp. 507–514, Ann Arbor, MI.

Collins, M., & Koo, T. (2005). Discriminative reranking for natural language parsing. *Computational Linguistics*, *31*(1), 25–69.

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

Darroch, J., & Ratchliff, D. (1972). Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, *43*(5), 1470–1480.

Della Pietra, S., Della Pietra, V. J., & Lafferty, J. D. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(4), 380–393.

Fletcher, R., & Reeves, C. M. (1964). Function minimization by conjugate gradients. *The Computer Journal*, *7*, 149–154.

Geman, S., & Johnson, M. (2002). Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 279–286, Philadelphia, PA.

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

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

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

Hockenmaier, J., & Steedman, M. (2002). Generative models for statistical parsing with combinatory categorial grammar. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 335–342, Philadelphia, PA.

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

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proc. of 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Canada.

Johnson, M., Geman, S., Canon, S., Chi, Z., & Riezler, S. (1999). Estimators for stochastic "unification-based" grammars. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, pp. 535–541, College Park, MD.

Jurafsky, D., & Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, Upper Saddle River, NJ.

Kate, R. J., Wong, Y. W., & Mooney, R. J. (2005). Learning to transform natural to formal languages. In *Proc. of 20th Natl. Conf. on Artificial Intelligence (AAAI-2005)*, pp. 1062–1068, Pittsburgh, PA.

Kate, R. J. (2005). A kernel-based approach to learning semantic parsers. Doctoral Dissertation Proposal, University of Texas at Austin.

Kuhlmann, G., Stone, P., Mooney, R., & Shavlik, J. (2004). Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *Proc. of the AAAI-04 Workshop on Supervisory Control of Learning and Adaptive Systems*, San Jose, CA.

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

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of 18th Intl. Conf. on Machine Learning (ICML-2001)*, pp. 282–289, Williamstown, MA.

Lesk, M. (1986). Automatic sense disambiguation: How to tell a pine cone from an ice cream cone. In *Special Interest Group on Design of Communication*.

Lev, I., MacCartney, B., Manning, C. D., & Levy, R. (2004). Solving logic puzzles: From robust processing to precise semantics. In *In proceedings of 2nd Workshop on Text Meaning and Interpretation, ACL-04*, Barcelona, Spain.

Levenshtein, V. I. (1966). Binary codes capable of correcting insertions and reversals. *Soviet Physics Doklady*, *10*(8), 707–710.

Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. In *Proc. of 6th Conf. on Computational Natural Language Learning (CoNLL-2002)*, pp. 49–55, Taipei, Taiwan.

Manning, C. D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

Marcus, M., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, *19*(2), 313–330.

McCallum, A., Freitag, D., & Pereira, F. (2000). Maximum entropy Markov models for information extraction and segmentation. In *Proc. of 17th Intl. Conf. on Machine Learning (ICML-2000)*, Stanford, CA.

Miller, G. (1991). WordNet: An on-line lexical database. *Intl. Journal of Lexicography*, *3*(4).

Miller, S., Bobrow, R., Ingria, R., & Schwartz, R. (1994). Hidden understanding models of natural language. In *Proc. of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL-94)*, pp. 25–32.

Miller, S., Fox, H., Ramshaw, L. A., & Weischedel, R. M. (2000). A novel use of statistical parsing to extract information from text. In *Proc. of the Meeting of the North American Association for Computational Linguistics*, pp. 226–233, Seattle, Washington.

Miller, S., Stallard, D., Bobrow, R., & Schwartz, R. (1996). A fully statistical approach to natural language interfaces. In *Proc. of the 34th Annual Meeting of the Association for Computational Linguistics (ACL-96)*, pp. 55–61, Santa Cruz, CA.

Miyao, Y., & Tsujii, J. (2002). Maximum entropy estimation for feature forests.. San Diego, CA.

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

Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, *39*, 103–134.

Nocedal, J., & Wright, S. J. (1999). *Numerical Optimization*. Springer Series in Operations Research.

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., Jain, V., Jin, Z., & Radev, D. (2004). A smorgasbord of features for statistical machine translation. In *Proc. of Human Language Technology Conf. / North American Association for Computational Linguistics Annual Meeting (HLT-NAACL-2004)*, pp. 161–168, Boston, MA.

Och, F. J., & Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 295–302, Philadelphia, PA.

Pook, S. L., & Catlett, J. (1988). Making sense out of searching. In *Proc. of the Online Information Conference*, pp. 69–78, Sydney, Australia.

Popescu, A.-M., Armanasu, A., Etzioni, O., Ko, D., & Yates, A. (2004). Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *Proc. of 20th Intl. Conf. on Computational Linguistics (COLING-04)*, Geneva, Switzerland.

Price, P. J. (1990). Evaluation of spoken language systems: The ATIS domain. In *Proc. of 3rd DARPA Speech and Natural Language Workshop*, pp. 91–95.

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

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

Riezler, S., King, T., Kaplan, R., Crouch, R., III, J. M., & Johnson, M. (2002). Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 271–278, Philadelphia, PA.

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

Seeger, M. (2000). Learning with labeled and unlabeled data. Tech. rep., Institute for ANC, Edinburgh, UK. See `http://www.dai.ed.ac.uk/~seeger/papers.html`.

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

Tang, L. R., & Mooney, R. J. (2001). Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proc. of the 12th European Conf. on Machine Learning*, pp. 466–477, Freiburg, Germany.

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

Wong, Y. W. (2005). Learning for semantic parsing using statistical machine translation techniques. Doctoral Dissertation Proposal, University of Texas at Austin.

Zelle, J. M., & Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proc. of 13th Natl. Conf. on Artificial Intelligence (AAAI-96)*, pp. 1050–1055, Portland, OR.

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

Zue, V. W., & Glass, J. R. (2000). Conversational interfaces: Advances and challenges. In *Proc. of the IEEE*, Vol. 88(8), pp. 1166–1180.