A Statistical Semantic Parser that Integrates Syntax and Semantics

Ruifang Ge Raymond J. Mooney
Department of Computer Sciences
University of Texas, Austin
TX 78712, USA
{grf,mooney}@cs.utexas.edu

Abstract

We introduce a learning semantic parser, SCISSOR, that maps natural-language sentences to a detailed, formal, meaning-representation language. It first uses an integrated statistical parser to produce a semantically augmented parse tree, in which each non-terminal node has both a syntactic and a semantic label. A compositional-semantics procedure is then used to map the augmented parse tree into a final meaning representation. We evaluate the system in 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.

1 Introduction

Most recent work in learning for semantic parsing has focused on "shallow" analysis such as *semantic role labeling* (Gildea and Jurafsky, 2002). In this paper, we address the more ambitious task of learning to map sentences to a complete formal *meaning-representation language* (MRL). We consider two MRL's that can be directly used to perform useful, complex tasks. The first is a Prolog-based language used in a previously-developed corpus of queries to a database on U.S. geography (Zelle and Mooney, 1996). The second MRL is a coaching language for robotic soccer developed for the RoboCup Coach Competition, in which AI researchers compete to provide effective instructions to a coachable team of agents in a simulated soccer domain (Chen et al., 2003).

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' head-driven model 2 (Collins, 1997) 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 an SAPT is generated, an additional step is required to translate it into a final formal *meaning representation* (MR).

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 SAPT's and MR's. We present experimental results on corpora for both geography-database querying and Robocup coaching demonstrating that SCISSOR produces more accurate semantic representations than several previous approaches based on symbolic learning (Tang and Mooney, 2001; Kate et al., 2005).

2 Target MRL's

We used two MRLs in our experiments: CLANG and GEOQUERY. They capture the meaning of linguistic utterances in their domain in a formal language.

2.1 CLANG: the RoboCup Coach Language

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."

2.2 GEOQUERY: a DB Query Language

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 and 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?"

3 Semantic Parsing Framework

This section describes our basic framework for semantic parsing, which is based on a fairly standard approach to compositional semantics (Jurafsky and Martin, 2000). First, a statistical parser is used to construct an 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

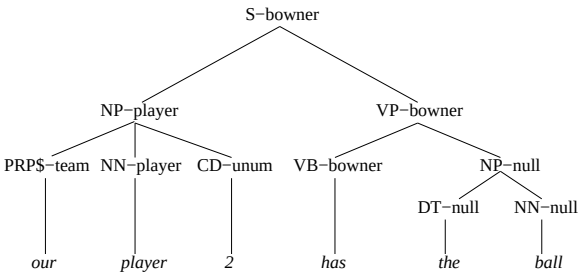

Figure 1: An SAPT for a simple CLANG sentence.

Function: BUILDMR(N, K)
Input: The root node N of an 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
$C_{h_{MR}} =$ 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
$N_{MR} = C_{h_{MR}}$

Figure 2: Computing an MR from an SAPT.

from the semantic label of the node and the MR's 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 an SAPT, we can use both semantic information to resolve syntactic ambiguities and syntactic information to resolve semantic ambiguities.

In an SAPT, each internal node in the parse tree is annotated with a semantic label. Figure 1 shows the SAPT for a simple sentence in the CLANG domain. 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* (ball owner). 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 2 shows the basic algorithm for build-

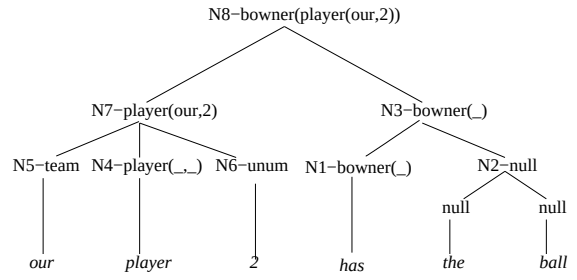

Figure 3: MR's constructed for each SAPT Node.

ing an MR from an SAPT. Figure 3 illustrates the construction of the MR for the SAPT in Figure 1. Nodes are numbered in the order in which the construction of their MR's 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 MR's 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 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 MRL's, 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.

There are a few complications left which require special handling when generating MR's, like coordinations and anaphora resolution. Due to space limitations, we do not present the straightforward techniques we used to handle them.

4 Corpus Annotation

This section discusses how sentences for training SCISSOR were manually annotated with SAPT's. Sentences were parsed by Collins' head-driven model 2 (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 augmented with a semantic label.

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

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 will inherit its label. The semantic head is chosen as the child whose semantic label can take the MR's of the other children as arguments. 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 kind 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 4 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.

The *point* label of the root node of Figure 4 is the concept that includes all kinds of points in CLANG. Once a predicate has all of its arguments filled, we

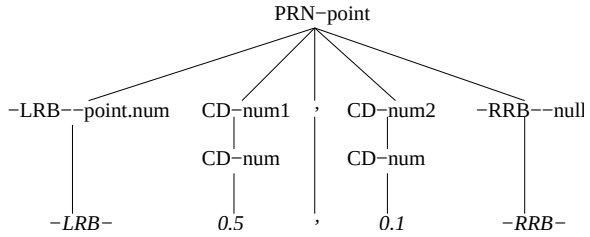

Figure 4: Adding new types to disambiguate arguments.

use the most general CLANG label for its concept (e.g. *point* instead of *point.num*). This generality avoids sparse data problems during training.

5 Integrated Parsing Model

In this section, we first summarize Collins' Model 2, and then present our extension to integrate semantics into it.

5.1 Collins Head-Driven Model 2

Collins' head-driven model 2 is a generative, lexicalized model of statistical parsing. In the following section, we follow the notation in (Collins, 1997). Each non-terminal X in the tree is a syntactic label, which is lexicalized by annotating it with a *word*, w, and a *POS tag*, t_{syn}. Thus, we write a non-terminal as $X(x)$, where X is a syntactic label and $x = \langle w, t_{syn} \rangle$. $X(x)$ is then what is generated by the generative model.

Each production $LHS \Rightarrow RHS$ in the PCFG is in the form:

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

where H is the head-child of the phrase, which inherits the head-word h from its parent P. $L_1...L_n$ and $R_1...R_m$ are left and right modifiers of H.

Sparse data makes the direct estimation of $\mathcal{P}(RHS|LHS)$ infeasible. Therefore, it is decomposed into several steps – first generating the head, then the right modifiers from the head outward, then the left modifiers in the same way. Syntactic subcategorization frames, LC and RC, for the left and right modifiers respectively, are generated before the generation of the modifiers. Subcat frames represent knowledge about subcategorization preferences. The final probability of a production is composed from the following probabilities:

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

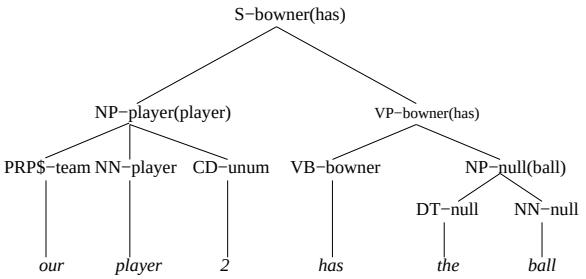

Figure 5: A lexicalized SAPT.

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 measure of the distance from the head word to the edge of the constituent, and $L_{n+1}(l_{n+1})$ and $R_{m+1}(r_{m+1})$ are $STOP$. The model stops generating more modifiers when $STOP$ is generated.

5.2 Integrating Semantics into the Model

We extend Collins' model to include the generation of semantic labels in the derivation tree. Unless otherwise stated, notation has the same meaning as in Section 5.1. The subscript syn refers to the syntactic part, and sem refers to the semantic part. We redefine X and x to include semantics, each non-terminal X is now a pair of a syntactic label X_{syn} and a semantic label X_{sem}. Besides being annotated with the word, w, and the POS tag, t_{syn}, X is also annotated with the semantic tag, t_{sem}, of the head child. Thus, $X(x)$ now consists of $X = \langle X_{syn}, X_{sem} \rangle$, and $x = \langle w, t_{syn}, t_{sem} \rangle$. Figure 5 shows a lexicalized SAPT (but omitting t_{syn} and t_{sem}).

Similar to the syntactic subcat frames, we also condition the generation of modifiers on semantic subcat frames. Semantic subcat frames give semantic subcategorization preferences; for example, *player* takes a *team* and a *unum*. Thus LC and RC are now: $\langle LC_{syn}, LC_{sem} \rangle$ and $\langle RC_{syn}, RC_{sem} \rangle$. $X(x)$ is generated as in Section 5.1, but using the new definitions of $X(x)$, LC and RC. The implementation of semantic subcat frames is similar to syntactic subcat frames. They are multisets speci-

fying the semantic labels which the head requires in its left or right modifiers.

As an example, the probability of generating the phrase "our player 2" using NP-player $\rightarrow$ PRP$-[team](our) NN-player CD-[unum](2) is (omitting only the distance measure):

$\mathcal{P}_h$(NN-[player]|NP-[player],player)$\times$

$\mathcal{P}_{lc}$($\langle${},{team}$\rangle$|NP-[player],player)$\times$

$\mathcal{P}_{rc}$($\langle${},{unum}$\rangle$|NP-[player],player)$\times$

$\mathcal{P}_l$(PRP$-[team](our)|NP-[player],player,$\langle${},{team}$\rangle$)$\times$

$\mathcal{P}_r$(CD-[unum](2)|NP-[player],player,$\langle${},{unum}$\rangle$)$\times$

$\mathcal{P}_l$(STOP|NP-[player],player,$\langle${},{}$\rangle$)$\times$

$\mathcal{P}_r$(STOP|NP-[player],player,$\langle${},{}$\rangle$)

5.3 Smoothing

Since the left and right modifiers are independently generated in the same way, we only discuss smoothing for the left side. Each probability estimation in the above generation steps is called a *parameter*. To reduce the risk of sparse data problems, the parameters are decomposed as follows:

$$\begin{aligned}
\mathcal{P}_h(H|C) &= \mathcal{P}_{h_{syn}}(H_{syn}|C) \times \\
&\quad \mathcal{P}_{h_{sem}}(H_{sem}|C, H_{syn}) \\
\mathcal{P}_{lc}(LC|C) &= \mathcal{P}_{lc_{syn}}(LC_{syn}|C) \times \\
&\quad \mathcal{P}_{lc_{sem}}(LC_{sem}|C, LC_{syn}) \\
\mathcal{P}_l(L_i(l_i)|C) &= \mathcal{P}_{l_{syn}}(L_{i_{syn}}(lt_{i_{syn}}, lw_i)|C) \times \\
&\quad \mathcal{P}_{l_{sem}}(L_{i_{sem}}(lt_{i_{sem}}, lw_i)|C, L_{i_{syn}}(lt_{i_{syn}}))
\end{aligned}$$

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

We make the independence assumption that the syntactic output is only conditioned on syntactic features, and semantic output on semantic ones. Note that the syntactic and semantic parameters are still integrated in the model to find the globally most likely parse. The syntactic parameters are the same as in Section 5.1 and are smoothed as in (Collins, 1997).

Since the semantic parameters do not depend on any syntactic features, we omit the sem subscripts in the following discussion. As in (Collins, 1997), the parameter $\mathcal{P}_l(L_i(lt_i, lw_i)|P, H, w, t, \Delta, LC)$ is further smoothed as follows:

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

Note this smoothing is different from the syntactic counterpart. This is due to the difference between POS tags and semantic tags; namely, semantic tags are generally more specific.

Table 1 shows the various levels of back-off for each semantic parameter. The probabilities from these back-off levels are interpolated using the techniques in (Collins, 1997). All words occurring less than 3 times in the training data, and words in test data that were not seen in training, are replaced with the "UNKNOWN" token. Note this threshold is smaller than the one used in (Collins, 1997) since the corpora used in our experiments are smaller.

5.4 POS Tagging and Semantic Tagging

For unknown words, gold-standard POS tags are used, otherwise they are generated along with the words using the same approach as in (Collins, 1997). When parsing, semantic tags for each known word are limited to those seen with that word during training data. The semantic tags allowed for an unknown word are limited to those seen with its associated POS tags during training.

6 Experimental Evaluation

6.1 Methodology

Two corpora of NL sentences paired with MR's were used to evaluate SCISSOR. For CLANG, 300 pieces of coaching advice were randomly selected from the log files of the 2003 RoboCup Coach Competition. Each formal instruction was translated into English by one of four annotators (Kate et al., 2005). The average length of an NL sentence in this corpus is 22.52 words. For GEOQUERY, 250 questions were collected by asking undergraduate students to generate English queries for the given database. Queries were then manually translated into logical form (Zelle and Mooney, 1996). The average length of an NL sentence in this corpus is 6.87 words. The queries in this corpus are more

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 for semantic parameters (sem subscripts omitted).

complex than those in the ATIS database-query corpus used in the speech community (Zue and Glass, 2000) which makes the GEOQUERY problem harder, as also shown by the results in (Popescu et al., 2004).

SCISSOR was evaluated using standard 10-fold cross validation. NL test sentences are first parsed to generate their SAPT's, then their MR's were built from the trees. We measured the number of test sentences that produced complete MR's, and the number of these MR's that were correct. For CLANG, an MR is correct if it exactly matches the correct representation, up to reordering of the arguments of commutative operators like `and`. For GEOQUERY, an MR is correct if 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* (the percentage of completed MR's that were correct) and *recall* (the percentage of all sentences whose MR's were correctly generated).

We compared SCISSOR's performance to several previous systems that learn semantic parsers that can map sentences into formal MRL's. CHILL (Zelle and Mooney, 1996) is a system based on Inductive Logic Programming (ILP). We compare to the version of CHILL presented in (Tang and Mooney, 2001), which uses the improved COCKTAIL ILP system and produces more accurate parsers than the original version presented in (Zelle and Mooney, 1996). SILT is a system that learns symbolic, pattern-based, transformation rules for mapping NL sentences to formal languages (Kate et al., 2005). It comes in two versions, SILT-string, which maps NL strings directly to an MRL, and SILT-tree, which maps syntactic parse trees (generated by the Collins parser) to an MRL. In the GEOQUERY domain, we also compare to the original hand-built parser GEOBASE.

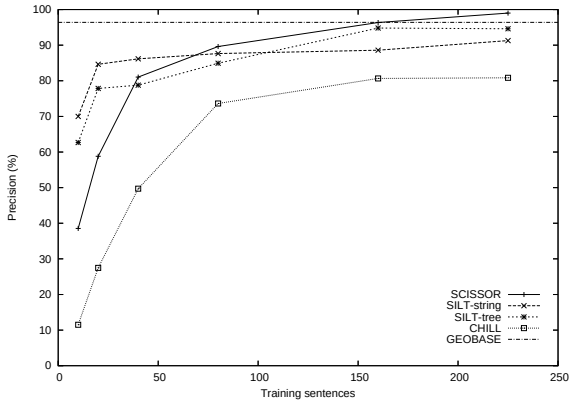

Figure 6: Precision learning curves for GEOQUERY.

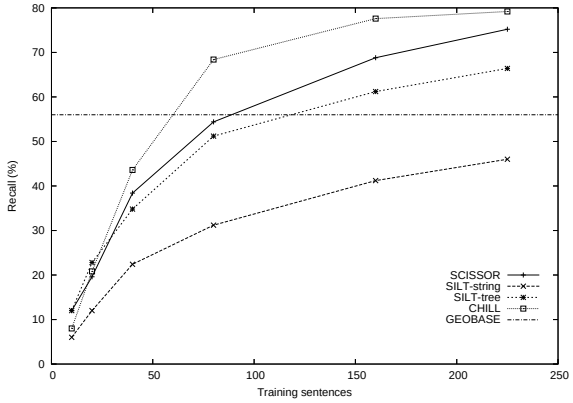

Figure 7: Recall learning curves for GEOQUERY.

6.2 Results

Figures 6 and 7 show the precision and recall learning curves for GEOQUERY, and Figures 8 and 9 for CLANG. Since CHILL is very memory intensive, it could not be run with larger training sets of the CLANG corpus.

Overall, SCISSOR gives the best precision and recall results in both domains. The only exception is with recall for GEOQUERY, for which CHILL is slightly higher. However, SCISSOR has significantly higher precision (see discussion in Section 7).

Results on a larger GEOQUERY corpus with 880 queries have been reported for PRECISE (Popescu et al., 2003): 100% precision and 77.5% recall. On

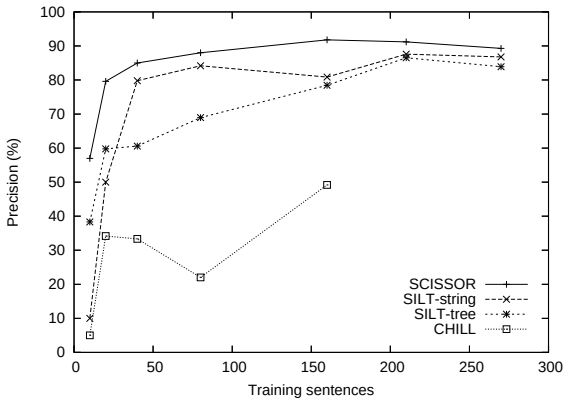

Figure 8: Precision learning curves for CLANG.

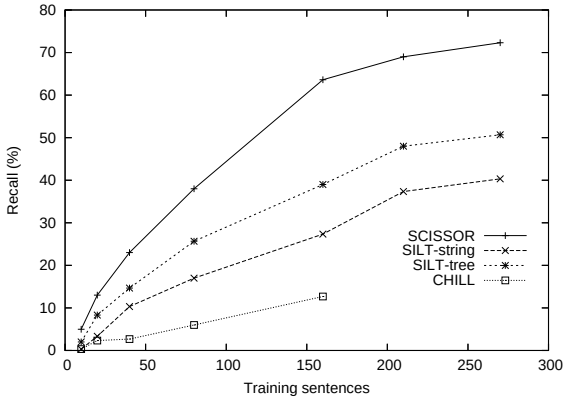

Figure 9: Recall learning curves for CLANG.

the same corpus, SCISSOR obtains 92.1% precision and 72.3% recall. However, the figures are not comparable. PRECISE can return multiple distinct SQL queries when it judges a question to be ambiguous and it is considered correct when *any* of these SQL queries is correct. Our measure only considers the top result. Due to space limitations, we do not present complete learning curves for this corpus.

7 Related Work

We first discuss the systems introduced in Section 6. CHILL uses computationally-complex ILP methods, which are slow and memory intensive. The string-based version of SILT uses no syntactic information while the tree-based version generates a syntactic parse first and then transforms it into an MR. In contrast, SCISSOR integrates syntactic and semantic processing, allowing each to constrain and inform the other. It uses a successful approach to statistical parsing that attempts to find the SAPT with maximum likelihood, which improves robustness compared to purely rule-based approaches. However, SCISSOR requires an extra training input, gold-standard SAPT's, not required by these other systems. Further automating the construction of training SAPT's from sentences paired with MR's is a subject of on-going research.

PRECISE is designed to work only for the specific task of NL database intrefaces. By comparison, SCISSOR is more general and can work with other MRL's as well (e.g. CLANG). Also, PRECISE is not a learning system and can fail to parse a query it considers ambiguous, even though it may not be considered ambiguous by a human and could potentially be resolved by learning regularities in the training data.

In (Lev et al., 2004), a syntax-driven approach is used to map logic puzzles described in NL to an MRL. The syntactic structures are paired with hand-written rules. A statistical parser is used to generate syntactic parse trees, and then MR's are built using compositional semantics. The meaning of open-category words (with only a few exceptions) is considered irrelevant to solving the puzzle and their meanings are not resolved. Further steps would be needed to generate MR's in other domains like CLANG and GEOQUERY. No empirical results are reported for their approach.

Several machine translation systems also attempt to generate MR's for sentences. In (Gao et al., 2002), an English-Chinese speech translation system for limited domains is described. They train a statistical parser on trees with only semantic labels on the nodes; however, they do not integrate syntactic and semantic parsing.

History-based models of parsing were first introduced in (Black et al., 1993). Their original model also included semantic labels on parse-tree nodes, but they were not used to generate a formal MR. Also, their parsing model is impoverished compared to the history included in Collins' more recent model. SCISSOR explores incorporating semantic labels into Collins' model in order to produce a complete SAPT which is then used to generate a formal MR.

The systems introduced in (Miller et al., 1996; Miller et al., 2000) also integrate semantic labels into parsing; however, their SAPT's are used to produce a much simpler MR, i.e., a single semantic frame. A sample frame is AIRTRANSPORTATION which has three slots – the arrival time, ori-

gin and destination. Only one frame needs to be extracted from each sentence, which is an easier task than our problem in which multiple nested frames (predicates) must extracted. The syntactic model in (Miller et al., 2000) is similar to Collins', but does not use features like subcat frames and distance measures. Also, the non-terminal label X is not further decomposed into separately-generated semantic and syntactic components. Since it used much more specific labels (the cross-product of the syntactic and semantic labels), its parameter estimates are potentially subject to much greater sparse-data problems.

8 Conclusion

SCISSOR learns statistical parsers that integrate syntax and semantics in order to produce a semantically augmented parse tree that is then used to compositionally generate a formal meaning representation. Experimental results in two domains, a natural-language database interface and an interpreter for coaching instructions in robotic soccer, have demonstrated that SCISSOR generally produces more accurate semantic representations than several previous approaches. By augmenting a state-of-the-art statistical parsing model to include semantic information, it is able to integrate syntactic and semantic clues to produce a robust interpretation that supports the generation of complete formal meaning representations.

References

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

Ezra Black, Frederick Jelineck, John Lafferty, David M. Magerman, Robert L. Mercer, and Salim Roukos. 1993. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proc. of ACL-93*, pages 31–37, Columbus, Ohio.

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

Mao Chen, Ehsan Foroughi, Fredrik Heintz, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummeneje, Itsuki Noda, Oliver Obst, Patrick Riley, Timo Steffens, Yi Wang, and Xiang Yin. 2003. Users manual: RoboCup soccer server manual for soccer server version 7.07 and later. Available at `http://sourceforge.net/projects/sserver/`.

Michael J. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of ACL-97*, pages 16–23, Madrid, Spain.

Yuqing Gao, Bowen Zhou, Zijian Diao, Jeffrey Sorensen, and Michael Picheny. 2002. Mars: A statistical semantic parsing and generation-based multilingual automatic translation system. *Machine Translation*, 17:185–212.

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

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, Upper Saddle River, NJ.

Rohit J. Kate, Yuk Wah Wong, Ruifang Ge, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. Available at `http://www.cs.utexas.edu/users/ml/papers/nlp-submitted-05.pdf`.

Iddo Lev, Bill MacCartney, Christopher D. Manning, and Roger Levy. 2004. Solving logic puzzles: From robust processing to precise semantics. In *In proc. of 2nd Workshop on Text Meaning and Interpretation, ACL-04*, Barcelona, Spain.

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *ACL-96*, pages 55–61, Santa Cruz, CA.

Scott Miller, Heidi Fox, Lance A. Ramshaw, and Ralph M. Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proc. of NAACL-00*, pages 226–233, Seattle, Washington.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proc. of IUI-2003*, pages 149–157, Miami, FL. ACM.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *COLING-04*, Geneva, Switzerland.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proc. of ECML-01*, pages 466–477, Freiburg, Germany.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proc. of AAAI-96*, pages 1050–1055, Portland, OR.

Victor W. Zue and James R. Glass. 2000. Conversational interfaces: Advances and challenges. In *Proc. of the IEEE*, volume 88(8), pages 1166–1180.