Learning Synchronous Grammars for Semantic Parsing with Lambda Calculus

Yuk Wah Wong and Raymond J. Mooney
Department of Computer Sciences
The University of Texas at Austin
{ywwong,mooney}@cs.utexas.edu

Abstract

This paper presents the first empirical results to our knowledge on learning synchronous grammars that generate logical forms. Using statistical machine translation techniques, a semantic parser based on a synchronous context-free grammar augmented with λ-operators is learned given a set of training sentences and their correct logical forms. The resulting parser is shown to be the best-performing system so far in a database query domain.

1 Introduction

Originally developed as a theory of compiling programming languages (Aho and Ullman, 1972), synchronous grammars have seen a surge of interest recently in the *statistical machine translation* (SMT) community as a way of formalizing syntax-based translation models between natural languages (NL). In generating multiple parse trees in a single derivation, synchronous grammars are ideal for modeling syntax-based translation because they describe not only the hierarchical structures of a sentence and its translation, but also the exact correspondence between their sub-parts. Among the grammar formalisms successfully put into use in syntax-based SMT are synchronous context-free grammars (SCFG) (Wu, 1997) and synchronous tree-substitution grammars (STSG) (Yamada and Knight, 2001). Both formalisms have led to SMT systems whose performance is state-of-the-art (Chiang, 2005; Galley et al., 2006).

Synchronous grammars have also been used in other NLP tasks, most notably *semantic parsing*, which is the construction of a complete, formal *meaning representation* (MR) of an NL sentence. In our previous work (Wong and Mooney, 2006), semantic parsing is cast as a machine translation task, where an SCFG is used to model the translation of an NL into a formal meaning-representation language (MRL). Our algorithm, WASP, uses statistical models developed for syntax-based SMT for lexical learning and parse disambiguation. The result is a robust semantic parser that gives good performance in various domains. More recently, we show that our SCFG-based parser can be inverted to produce a state-of-the-art NL generator, where a formal MRL is translated into an NL (Wong and Mooney, 2007).

Currently, the use of learned synchronous grammars in semantic parsing and NL generation is limited to simple MRLs that are free of logical variables. This is because grammar formalisms such as SCFG do not have a principled mechanism for handling logical variables. This is unfortunate because most existing work on computational semantics is based on predicate logic, where logical variables play an important role (Blackburn and Bos, 2005). For some domains, this problem can be avoided by transforming a logical language into a variable-free, functional language (e.g. the GEOQUERY functional query language in Wong and Mooney (2006)). However, development of such a functional language is non-trivial, and as we will see, logical languages can be more appropriate for certain domains.

On the other hand, most existing methods for mapping NL sentences to logical forms involve substantial hand-written components that are difficult to maintain (Joshi and Vijay-Shanker, 2001; Bayer et al., 2004; Bos, 2005). Zettlemoyer and Collins (2005) present a statistical method that is consider-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 960–967,
Prague, Czech Republic, June 2007.

ably more robust, but it still relies on hand-written rules for lexical acquisition, which can create a performance bottleneck.

In this work, we show that methods developed for SMT can be brought to bear on tasks where logical forms are involved, such as semantic parsing. In particular, we extend the WASP semantic parsing algorithm by adding variable-binding λ-operators to the underlying SCFG. The resulting synchronous grammar generates logical forms using λ-calculus (Montague, 1970). A semantic parser is learned given a set of sentences and their correct logical forms using SMT methods. The new algorithm is called λ-WASP, and is shown to be the best-performing system so far in the GEOQUERY domain.

2 Test Domain

In this work, we mainly consider the GEOQUERY domain, where a query language based on Prolog is used to query a database on U.S. geography (Zelle and Mooney, 1996). The query language consists of logical forms augmented with meta-predicates for concepts such as *smallest* and *count*. Figure 1 shows two sample logical forms and their English glosses. Throughout this paper, we use the notation $x_1, x_2, \ldots$ for logical variables.

Although Prolog logical forms are the main focus of this paper, our algorithm makes minimal assumptions about the target MRL. The only restriction on the MRL is that it be defined by an unambiguous context-free grammar (CFG) that divides a logical form into subformulas (and terms into subterms). Figure 2(a) shows a sample parse tree of a logical form, where each CFG production corresponds to a subformula.

3 The Semantic Parsing Algorithm

Our work is based on the WASP semantic parsing algorithm (Wong and Mooney, 2006), which translates NL sentences into MRs using an SCFG. In WASP, each SCFG production has the following form:

$$A \rightarrow \langle \alpha, \beta \rangle \tag{1}$$

where α is an NL phrase and β is the MR translation of α. Both α and β are strings of terminal and non-terminal symbols. Each non-terminal in α appears in β exactly once. We use indices to show the correspondence between non-terminals in α and β. All derivations start with a pair of co-indexed start symbols, $\langle S_{\boxed{1}}, S_{\boxed{1}} \rangle$. Each step of a derivation involves the rewriting of a pair of co-indexed non-terminals by the same SCFG production. The yield of a derivation is a pair of terminal strings, $\langle \mathbf{e}, \mathbf{f} \rangle$, where $\mathbf{e}$ is an NL sentence and $\mathbf{f}$ is the MR translation of $\mathbf{e}$. For convenience, we call an SCFG production a *rule* throughout this paper.

While WASP works well for target MRLs that are free of logical variables such as CLANG (Wong and Mooney, 2006), it cannot easily handle various kinds of logical forms used in computational semantics, such as predicate logic. The problem is that WASP lacks a principled mechanism for handling logical variables. In this work, we extend the WASP algorithm by adding a variable-binding mechanism based on λ-calculus, which allows for compositional semantics for logical forms.

This work is based on an extended version of SCFG, which we call λ-SCFG, where each rule has the following form:

$$A \rightarrow \langle \alpha, \lambda x_1 \ldots \lambda x_k . \beta \rangle \tag{2}$$

where α is an NL phrase and β is the MR translation of α. Unlike (1), β is a string of terminals, non-terminals, and *logical variables*. The variable-binding operator λ binds occurrences of the logical variables $x_1, \ldots, x_k$ in β, which makes $\lambda x_1 \ldots \lambda x_k . \beta$ a λ-*function* of arity k. When applied to a list of arguments, $(x_{i_1}, \ldots, x_{i_k})$, the λ-function gives $\beta\sigma$, where σ is a substitution operator, $\{x_1/x_{i_1}, \ldots, x_k/x_{i_k}\}$, that replaces all bound occurrences of x_j in β with x_{i_j}. If any of the arguments x_{i_j} appear in β as a free variable (i.e. not bound by any λ), then those free variables in β must be renamed before function application takes place.

Each non-terminal A_j in β is followed by a list of arguments, $\mathbf{x}_j = (x_{j_1}, \ldots, x_{j_{k_j}})$. During parsing, A_j must be rewritten by a λ-function f_j of arity k_j. Like SCFG, a derivation starts with a pair of co-indexed start symbols and ends when all non-terminals have been rewritten. To compute the yield of a derivation, each f_j is applied to its corresponding arguments $\mathbf{x}_j$ to obtain an MR string free of λ-operators with logical variables properly named.

(a) $\texttt{answer(}x_1\texttt{,smallest(}x_2\texttt{,(state(}x_1\texttt{),area(}x_1\texttt{,}x_2\texttt{))))}$
What is the smallest state by area?

(b) $\texttt{answer(}x_1\texttt{,count(}x_2\texttt{,(city(}x_2\texttt{),major(}x_2\texttt{),loc(}x_2\texttt{,}x_3\texttt{),next\_to(}x_3\texttt{,}x_4\texttt{),state(}x_3\texttt{),}$
$\texttt{equal(}x_4\texttt{,stateid(texas)))))}$
How many major cities are in states bordering Texas?

Figure 1: Sample logical forms in the GEOQUERY domain and their English glosses.

(a)
QUERY
$\texttt{answer(}x_1\texttt{,}$FORM$\texttt{)}$
$\texttt{smallest(}x_2\texttt{,(}$FORM$\texttt{,}$FORM$\texttt{))}$
$\texttt{state(}x_1\texttt{)}$ $\texttt{area(}x_1\texttt{,}x_2\texttt{)}$

(b)
QUERY
$\texttt{answer(}x_1\texttt{,}$FORM$(x_1)\texttt{)}$
$\lambda x_1.\texttt{smallest(}x_2\texttt{,(}$FORM$(x_1)\texttt{,}$FORM$(x_1,x_2)\texttt{))}$
$\lambda x_1.\texttt{state(}x_1\texttt{)}$ $\lambda x_1.\lambda x_2.\texttt{area(}x_1\texttt{,}x_2\texttt{)}$

Figure 2: Parse trees of the logical form in Figure 1(a).

As a concrete example, Figure 2(b) shows an MR parse tree that corresponds to the English parse, [*What is the* [*smallest* [*state*] [*by area*]]], based on the λ-SCFG rules in Figure 3. To compute the yield of this MR parse tree, we start from the leaf nodes: apply $\lambda x_1.\texttt{state(}x_1\texttt{)}$ to the argument (x_1), and $\lambda x_1.\lambda x_2.\texttt{area(}x_1\texttt{,}x_2\texttt{)}$ to the arguments (x_1, x_2). This results in two MR strings: $\texttt{state(}x_1\texttt{)}$ and $\texttt{area(}x_1\texttt{,}x_2\texttt{)}$. Substituting these MR strings for the FORM non-terminals in the parent node gives the λ-function $\lambda x_1.\texttt{smallest(}x_2\texttt{,(state(}x_1\texttt{),area(}x_1\texttt{,}x_2\texttt{)))}$. Applying this λ-function to (x_1) gives the MR string $\texttt{smallest(}x_2\texttt{,(state(}x_1\texttt{),area(}x_1\texttt{,}x_2\texttt{)))}$. Substituting this MR string for the FORM non-terminal in the grandparent node in turn gives the logical form in Figure 1(a). This is the yield of the MR parse tree, since the root node of the parse tree is reached.

3.1 Lexical Acquisition

Given a set of training sentences paired with their correct logical forms, $\{\langle \mathbf{e}_i, \mathbf{f}_i \rangle\}$, the main learning task is to find a λ-SCFG, G, that covers the training data. Like most existing work on syntax-based SMT (Chiang, 2005; Galley et al., 2006), we construct G using rules extracted from word alignments. We use the $K = 5$ most probable word alignments for the training set given by GIZA++ (Och and Ney, 2003), with variable names ignored to reduce sparsity. Rules are then extracted from each word alignment as follows.

To ground our discussion, we use the word alignment in Figure 4 as an example. To represent the logical form in Figure 4, we use its linearized parse—a list of MRL productions that generate the logical form, in top-down, left-most order (cf. Figure 2(a)). Since the MRL grammar is unambiguous, every logical form has a unique linearized parse. We assume the alignment to be n-to-1, where each word is linked to at most one MRL production.

Rules are extracted in a bottom-up manner, starting with MRL productions at the leaves of the MR parse tree, e.g. FORM $\rightarrow \texttt{state(}x_1\texttt{)}$ in Figure 2(a). Given an MRL production, $A \rightarrow \beta$, a rule $A \rightarrow \langle \alpha, \lambda x_{i_1} \ldots \lambda x_{i_k}.\beta \rangle$ is extracted such that: (1) α is the NL phrase linked to the MRL production; (2) $x_{i_1}, \ldots, x_{i_k}$ are the logical variables that appear in β and outside the current leaf node in the MR parse tree. If $x_{i_1}, \ldots, x_{i_k}$ were not bound by λ, they would become free variables in β, subject to renaming during function application (and therefore, invisible to the rest of the logical form). For example, since x_1 is an argument of the $\texttt{state}$ predicate as well as $\texttt{answer}$ and $\texttt{area}$, x_1 must be bound (cf. the corresponding tree node in Figure 2(b)). The rule extracted for the $\texttt{state}$ predicate is shown in Figure 3.

The case for the internal nodes of the MR parse tree is similar. Given an MRL production, $A \rightarrow \beta$, where β contains non-terminals $A_1, \ldots, A_n$, a rule $A \rightarrow \langle \alpha, \lambda x_{i_1} \ldots \lambda x_{i_k}.\beta' \rangle$ is extracted such that: (1) α is the NL phrase linked to the MRL production, with non-terminals $A_1, \ldots, A_n$ showing the positions of the argument strings; (2) β' is β with each non-terminal A_j replaced with $A_j(x_{j_1}, \ldots, x_{j_{k_j}})$, where $x_{j_1}, \ldots, x_{j_{k_j}}$ are the bound variables in the λ-function used to rewrite A_j; (3) $x_{i_1}, \ldots, x_{i_k}$ are the logical variables that appear in β' and outside the current MR sub-parse. For example, see the rule

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

Figure 3: λ-SCFG rules for parsing the English sentence in Figure 1(a).

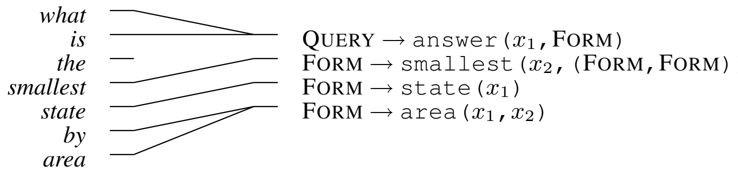


Figure 4: Word alignment for the sentence pair in Figure 1(a).

extracted for the `smallest` predicate in Figure 3, where x_2 is an argument of `smallest`, but it does not appear outside the formula `smallest(...)`, so x_2 need not be bound by λ. On the other hand, x_1 appears in β', and it appears outside `smallest(...)` (as an argument of `answer`), so x_1 must be bound.

Rule extraction continues in this manner until the root of the MR parse tree is reached. Figure 3 shows all the rules extracted from Figure 4.[1]

3.2 Probabilistic Semantic Parsing Model

Since the learned λ-SCFG can be ambiguous, a probabilistic model is needed for parse disambiguation. We use the maximum-entropy model proposed in Wong and Mooney (2006), which defines a conditional probability distribution over derivations given an observed NL sentence. The output MR is the yield of the most probable derivation according to this model.

Parameter estimation involves maximizing the conditional log-likelihood of the training set. For each rule, r, there is a feature that returns the number of times r is used in a derivation. More features will be introduced in Section 5.

4 Promoting NL/MRL Isomorphism

We have described the λ-WASP algorithm which generates logical forms based on λ-calculus. While reasonably effective, it can be improved in several ways. In this section, we focus on improving lexical acquisition.

To see why the current lexical acquisition algorithm can be problematic, consider the word alignment in Figure 5 (for the sentence pair in Figure 1(b)). No rules can be extracted for the `state` predicate, because the shortest NL substring that covers the word *states* and the argument string *Texas*, i.e. *states bordering Texas*, contains the word *bordering*, which is linked to an MRL production outside the MR sub-parse rooted at `state`. Rule extraction is forbidden in this case because it would destroy the link between *bordering* and `next_to`. In other words, the NL and MR parse trees are not *isomorphic*.

This problem can be ameliorated by transforming the logical form of each training sentence so that the NL and MR parse trees are maximally isomorphic. This is possible because some of the operators used in the logical forms, notably the conjunction operator (`,`), are both associative (`a,(b,c)` = `(a,b),c` = `a,b,c`) and commutative (`a,b` = `b,a`). Hence, conjuncts can be reordered and regrouped without changing the meaning of a conjunction. For example, rule extraction would be possible if the positions of the `next_to` and `state` conjuncts were switched. We present a method for regrouping conjuncts to promote isomorphism between NL and MR parse trees.[2] Given a conjunction, it does the following: (See Figure 6 for the pseudocode, and Figure 5 for an illustration.)

Step 1. Identify the MRL productions that correspond to the conjuncts and the meta-predicate that takes the conjunction as an argument (`count` in Figure 5), and figure them as vertices in an undi-

[1]For details regarding non-isomorphic NL/MR parse trees, removal of bad links from alignments, and extraction of word gaps (e.g. the token (1) in the last rule of Figure 3), see Wong and Mooney (2006).

[2]This method also applies to *any* operators that are associative and commutative, e.g. disjunction. For concreteness, however, we use conjunction as an example.

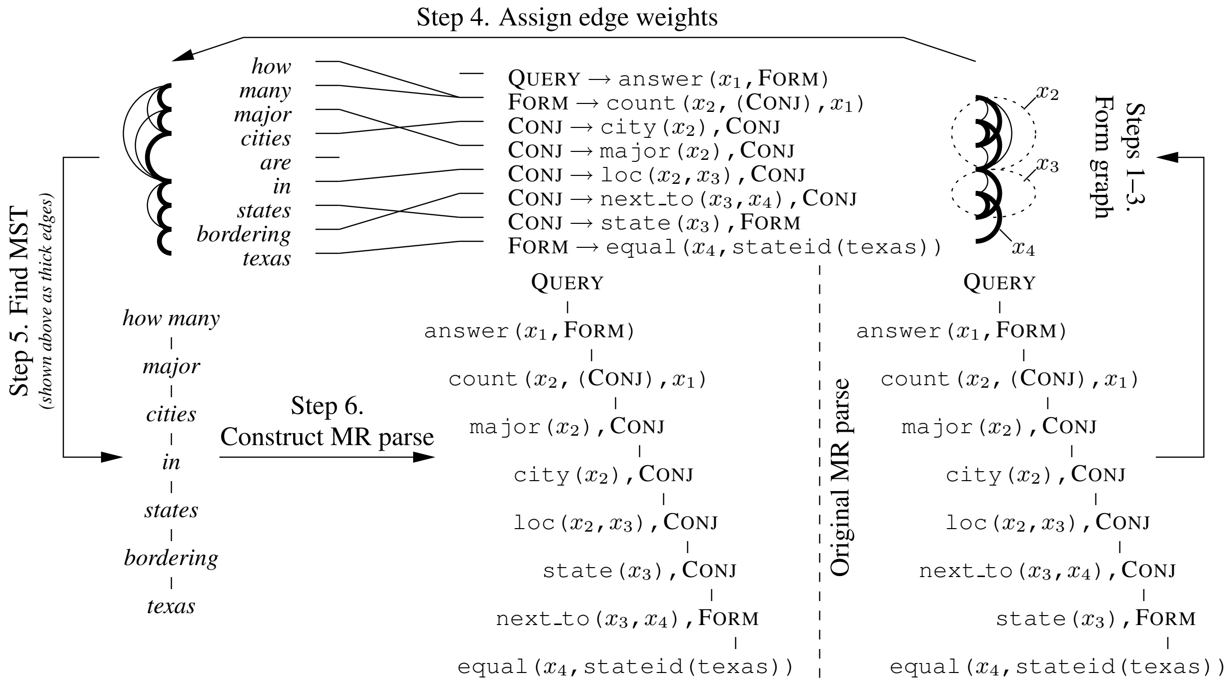


Figure 5: Transforming the logical form in Figure 1(b). The step numbers correspond to those in Figure 6.

Input: A conjunction, $\mathbf{c}$, of n conjuncts; MRL productions, $p_1, \ldots, p_n$, that correspond to each conjunct; an MRL production, p_0, that corresponds to the meta-predicate taking $\mathbf{c}$ as an argument; an NL sentence, $\mathbf{e}$; a word alignment, $\mathbf{a}$.

1 Let $\mathbf{v}(p)$ be the set of logical variables that appear in p. Create an undirected graph, Γ, with vertices $V = \{p_i | i = 0, \ldots, n\}$ and edges $E = \{(p_i, p_j) | i < j, \mathbf{v}(p_i) \cap \mathbf{v}(p_j) \neq \emptyset\}$.
2 Let $\mathbf{e}(p)$ be the set of words in $\mathbf{e}$ to which p is linked according to $\mathbf{a}$. Let $\mathrm{span}(p_i, p_j)$ be the shortest substring of $\mathbf{e}$ that includes $\mathbf{e}(p_i) \cup \mathbf{e}(p_j)$. Subtract $\{(p_i, p_j) | i \neq 0, \mathrm{span}(p_i, p_j) \cap \mathbf{e}(p_0) \neq \emptyset\}$ from E.
3 Add edges (p_0, p_i) to E if p_i is not already connected to p_0.
4 For each edge (p_i, p_j) in E, set edge weight to the minimum word distance between $\mathbf{e}(p_i)$ and $\mathbf{e}(p_j)$.
5 Find a minimum spanning tree, T, for Γ using Kruskal's algorithm.
6 Using p_0 as the root, construct a conjunction $\mathbf{c}'$ based on T, and then replace $\mathbf{c}$ with $\mathbf{c}'$.

Figure 6: Algorithm for regrouping conjuncts to promote isomorphism between NL and MR parse trees.

rected graph, Γ. An edge (p_i, p_j) is in Γ if and only if p_i and p_j contain occurrences of the same logical variables. Each edge in Γ indicates a possible edge in the transformed MR parse tree. Intuitively, two concepts are closely related if they involve the same logical variables, and therefore, should be placed close together in the MR parse tree. By keeping occurrences of a logical variable in close proximity in the MR parse tree, we also avoid unnecessary variable bindings in the extracted rules.

Step 2. Remove edges from Γ whose inclusion in the MR parse tree would prevent the NL and MR parse trees from being isomorphic.

Step 3. Add edges to Γ to make sure that a spanning tree for Γ exists.

Steps 4–6. Assign edge weights based on word distance, find a minimum spanning tree, T, for Γ, then regroup the conjuncts based on T. The choice of T reflects the intuition that words that occur close together in a sentence tend to be semantically related.

This procedure is repeated for all conjunctions that appear in a logical form. Rules are then extracted from the same input alignment used to regroup conjuncts. Of course, the regrouping of conjuncts requires a good alignment to begin with, and that requires a reasonable ordering of conjuncts in the training data, since the alignment model is sensitive to word order. This suggests an iterative algorithm in which a better grouping of conjuncts leads to a better alignment model, which guides further regrouping until convergence. We did not pursue this, as it is not needed in our experiments so far.

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

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

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

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

Figure 7: Typical errors made by the λ-WASP parser, along with their English interpretations, before any language modeling for the target MRL was done.

5 Modeling the Target MRL

In this section, we propose two methods for modeling the target MRL. This is motivated by the fact that many of the errors made by the λ-WASP parser can be detected by inspecting the MR translations alone. Figure 7 shows some typical errors, which can be classified into two broad categories:

1. Type mismatch errors. For example, a state cannot possibly be a river (Figure 7(a)). Also it is awkward to talk about the population density of a state's highest point (Figure 7(b)).

2. Errors that do not involve type mismatch. For example, a query can be overly trivial (Figure 7(c)), or involve aggregate functions on a known singleton (Figure 7(d)).

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

$$\texttt{point(\_)}: \{(\textsc{point})\}$$
$$\texttt{density(\_,\_)}: \{(\textsc{country}, \textsc{num}), (\textsc{state}, \textsc{num}), (\textsc{city}, \textsc{num})\}$$

These m-tuples of entity types are given as domain knowledge. The parser maintains a set of possible entity types for each logical variable introduced in a partial derivation (except those that are no longer visible). If there is a logical variable that cannot refer to any types of entities (i.e. the set of entity types is empty), then the partial derivation is considered invalid. For example, based on the tuples shown above, `point(`x_1`)` and `density(`x_1`,_)` cannot be both true, because $\{\textsc{point}\} \cap \{\textsc{country}, \textsc{state}, \textsc{city}\} = \emptyset$. The use of type checking is to exploit the fact that people tend not to ask questions that obviously have no valid answers (Grice, 1975). It is also similar to Schuler's (2003) use of model-theoretic interpretations to guide syntactic parsing.

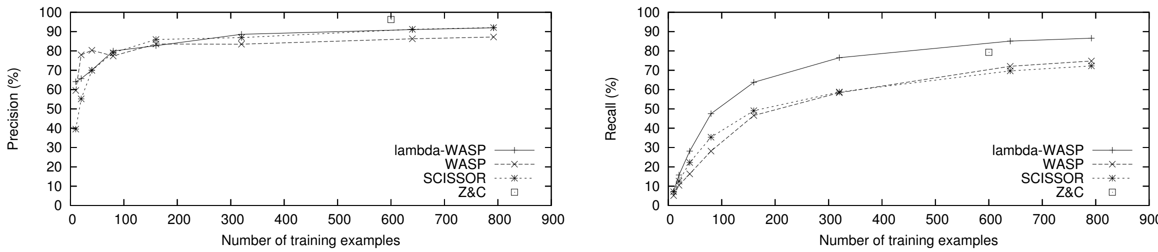
Errors that do not involve type mismatch are handled by adding new features to the maximum-entropy model (Section 3.2). We only consider features that are based on the MR translations, and therefore, these features can be seen as an implicit language model of the target MRL (Papineni et al., 1997). Of the many features that we have tried, one feature set stands out as being the most effective, the *two-level rules* in Collins and Koo (2005), which give the number of times a given rule is used to expand a non-terminal in a given parent rule. We use only the MRL part of the rules. For example, a negative weight for the combination of QUERY $\rightarrow$ `answer(`x_1`,`FORM(x_1)`)` and FORM $\rightarrow \lambda x_1$`.equal(`x_1`,_)` would discourage any parse that yields Figure 7(c). The two-level rules features, along with the features described in Section 3.2, are used in the final version of λ-WASP.

6 Experiments

We evaluated the λ-WASP algorithm in the GEOQUERY domain. The larger GEOQUERY corpus consists of 880 English questions gathered from various sources (Wong and Mooney, 2006). The questions were manually translated into Prolog logical forms. The average length of a sentence is 7.57 words.

We performed a single run of 10-fold cross validation, and measured the performance of the learned parsers using *precision* (percentage of translations that were correct), *recall* (percentage of test sentences that were correctly translated), and *F-measure* (harmonic mean of precision and recall). A translation is considered correct if it retrieves the same answer as the correct logical form.

Figure 8 shows the learning curves for the λ-



(a) Precision (b) Recall

Figure 8: Learning curves for various parsing algorithms on the larger GEOQUERY corpus.

(%)	λ-WASP	WASP	SCISSOR	Z&C
Precision	91.95	87.19	92.08	**96.25**
Recall	**86.59**	74.77	72.27	79.29
F-measure	**89.19**	80.50	80.98	86.95

Table 1: Performance of various parsing algorithms on the larger GEOQUERY corpus.

WASP algorithm compared to: (1) the original WASP algorithm which uses a functional query language (FunQL); (2) SCISSOR (Ge and Mooney, 2005), a fully-supervised, combined syntactic-semantic parsing algorithm which also uses FunQL; and (3) Zettlemoyer and Collins (2005) (Z&C), a CCG-based algorithm which uses Prolog logical forms. Table 1 summarizes the results at the end of the learning curves (792 training examples for λ-WASP, WASP and SCISSOR, 600 for Z&C).

A few observations can be made. First, algorithms that use Prolog logical forms as the target MRL generally show better recall than those using FunQL. In particular, λ-WASP has the best recall by far. One reason is that it allows lexical items to be combined in ways not allowed by FunQL or the hand-written templates in Z&C, e.g. [*smallest* [*state*] [*by area*]] in Figure 3. Second, Z&C has the best precision, although their results are based on 280 test examples only, whereas our results are based on 10-fold cross validation. Third, λ-WASP has the best F-measure.

To see the relative importance of each component of the λ-WASP algorithm, we performed two ablation studies. First, we compared the performance of λ-WASP with and without conjunct regrouping (Section 4). Second, we compared the performance of λ-WASP with and without language modeling for the MRL (Section 5). Table 2 shows the results. It is found that conjunct regrouping improves recall ($p < 0.01$ based on the paired t-test), and the use of *two-level rules* in the maximum-entropy model improves precision and recall ($p < 0.05$). Type checking also significantly improves precision and recall.

A major advantage of λ-WASP over SCISSOR and Z&C is that it does not require any prior knowledge of the NL syntax. Figure 9 shows the performance of λ-WASP on the multilingual GEOQUERY data set. The 250-example data set is a subset of the larger GEOQUERY corpus. All English questions in this data set were manually translated into Spanish, Japanese and Turkish, while the corresponding Prolog queries remain unchanged. Figure 9 shows that λ-WASP performed comparably for all NLs. In contrast, SCISSOR cannot be used directly on the non-English data, because syntactic annotations are only available in English. Z&C cannot be used directly either, because it requires NL-specific templates for building CCG grammars.

7 Conclusions

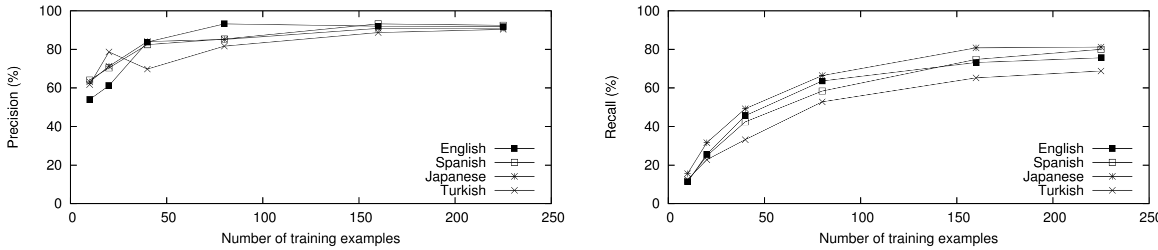
We have presented λ-WASP, a semantic parsing algorithm based on a λ-SCFG that generates logical forms using λ-calculus. A semantic parser is learned given a set of training sentences and their correct logical forms using standard SMT techniques. The result is a robust semantic parser for predicate logic, and it is the best-performing system so far in the GEOQUERY domain.

This work shows that it is possible to use standard SMT methods in tasks where logical forms are involved. For example, it should be straightforward to adapt λ-WASP to the NL generation task—all one needs is a decoder that can handle input logical forms. Other tasks that can potentially benefit from

(%)	Precision	Recall
λ-WASP	**91.95**	**86.59**
w/o conj. regrouping	**90.73**	83.07

(%)	Precision	Recall
λ-WASP	**91.95**	**86.59**
w/o two-level rules	88.46	84.32
and w/o type checking	65.45	63.18

Table 2: Performance of λ-WASP with certain components of the algorithm removed.



(a) Precision (b) Recall

Figure 9: Learning curves for λ-WASP on the multilingual GEOQUERY data set.

this include question answering and interlingual MT.

In future work, we plan to further generalize the synchronous parsing framework to allow different combinations of grammar formalisms. For example, to handle long-distance dependencies that occur in open-domain text, CCG and TAG would be more appropriate than CFG. Certain applications may require different meaning representations, e.g. frame semantics.

Acknowledgments: We thank Rohit Kate, Razvan Bunescu and the anonymous reviewers for their valuable comments. This work was supported by a gift from Google Inc.

References

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

S. Bayer, J. Burger, W. Greiff, and B. Wellner. 2004. The MITRE logical form generation system. In *Proc. of Senseval-3*, Barcelona, Spain, July.

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

J. Bos. 2005. Towards wide-coverage semantic interpretation. In *Proc. of IWCS-05*, Tilburg, The Netherlands, January.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL-05*, pages 263–270, Ann Arbor, MI, June.

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

M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, and I. Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of COLING/ACL-06*, pages 961–968, Sydney, Australia, July.

R. Ge and R. J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proc. of CoNLL-05*, pages 9–16, Ann Arbor, MI, July.

H. P. Grice. 1975. Logic and conversation. In P. Cole and J. Morgan, eds., *Syntax and Semantics 3: Speech Acts*, pages 41–58. Academic Press, New York.

A. K. Joshi and K. Vijay-Shanker. 2001. Compositional semantics with lexicalized tree-adjoining grammar (LTAG): How much underspecification is necessary? In H. Bunt et al., eds., *Computing Meaning*, volume 2, pages 147–163. Kluwer Academic Publishers, Dordrecht, The Netherlands.

R. Montague. 1970. Universal grammar. *Theoria*, 36:373–398.

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

K. A. Papineni, S. Roukos, and R. T. Ward. 1997. Feature-based language understanding. In *Proc. of EuroSpeech-97*, pages 1435–1438, Rhodes, Greece.

W. Schuler. 2003. Using model-theoretic semantic interpretation to guide statistical parsing and word recognition in a spoken language interface. In *Proc. of ACL-03*, pages 529–536.

Y. W. Wong and R. J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proc. of HLT/NAACL-06*, pages 439–446, New York City, NY.

Y. W. Wong and R. J. Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Proc. of NAACL/HLT-07*, Rochester, NY, to appear.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *Proc. of ACL-01*, pages 523–530, Toulouse, France.

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

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of UAI-05*, Edinburgh, Scotland, July.