Also appears as Technical Report UT-AI-05-323, Artificial Intelligence Lab, University of Texas at Austin, October 2005.

Learning for Semantic Parsing Using Statistical Machine Translation Techniques

Yuk Wah Wong
Department of Computer Sciences
The University of Texas at Austin
1 University Station C0500
Austin, TX 78712-0233, USA
ywwong@cs.utexas.edu

Doctoral Dissertation Proposal

Supervising Professor: Raymond J. Mooney

Abstract

Semantic parsing is the construction of a complete, formal, symbolic meaning representation of a sentence. While it is crucial to natural language understanding, the problem of semantic parsing has received relatively little attention from the machine learning community. Recent work on natural language understanding has mainly focused on shallow semantic analysis, such as word-sense disambiguation and semantic role labeling. Semantic parsing, on the other hand, involves deep semantic analysis in which word senses, semantic roles and other components are combined to produce useful meaning representations for a particular application domain (e.g. database query). Prior research in machine learning for semantic parsing is mainly based on inductive logic programming or deterministic parsing, which lack some of the robustness that characterizes statistical learning. Existing statistical approaches to semantic parsing, however, are mostly concerned with relatively simple application domains in which a meaning representation is no more than a single semantic frame.

In this proposal, we present a novel statistical approach to semantic parsing, WASP, which can handle meaning representations with a nested structure. The WASP algorithm learns a semantic parser given a set of sentences annotated with their correct meaning representations. The parsing model is based on the synchronous context-free grammar, where each rule maps a natural-language substring to its meaning representation. The main innovation of the algorithm is its use of state-of-the-art statistical machine translation techniques. A statistical word alignment model is used for lexical acquisition, and the parsing model itself can be seen as an instance of a syntax-based translation model. In initial evaluation on several real-world data sets, we show that WASP performs favorably in terms of both accuracy and coverage compared to existing learning methods requiring similar amount of supervision, and shows better robustness to variations in task complexity and word order.

In future work, we intend to pursue several directions in developing accurate semantic parsers for a variety of application domains. This will involve exploiting prior knowledge about the natural-language syntax and the application domain. We also plan to construct a syntax-aware word-based alignment model for lexical acquisition. Finally, we will generalize the learning algorithm to handle context-dependent sentences and accept noisy training data.

Contents

1 Introduction

Most current research in machine learning for natural language processing (NLP) has been for fairly low-level tasks such as part-of-speech tagging, word-sense disambiguation, and syntactic parsing (Manning & Schütze, 1999). The holy grail of natural language processing, however, is the construction of an automated agent that is able to understand human languages, communicate with humans via natural language generation, and do useful tasks through inference. Recent work on natural language understanding has mainly focused on shallow semantic analysis, such as *semantic role labeling* (Carreras & Màrquez, 2005; Koomen et al., 2005; Toutanova et al., 2005), which is concerned with finding phrases that fill in the semantic roles of a single predicate given a natural-language (NL) sentence. This proposal considers a more ambitious task of *semantic parsing*, which is the construction of a complete, formal, symbolic, *meaning representation* (MR) of a sentence. Semantic parsing has found its way in many practical applications such as natural-language interfaces (NLI) to databases (Hendrix et al., 1978; Price, 1990; Androutsopoulos et al., 1995), question answering (Friedland et al., 2004; Lev et al., 2004), command and control (Bellegarda & Silverman, 2003; Simmons et al., 2003), and advice taking (Kuhlmann et al., 2004).

Prior research in semantic parsing has mainly focused on relatively simple domains such as ATIS (Air Travel Information Service) (Price, 1990; Kuhn & De Mori, 1995; Miller et al., 1996; He & Young, 2003; Popescu et al., 2004), in which an MR is no more than a non-recursive semantic frame. Learning methods have been devised that can generate MRs with a more complex, nested structure, but these are mostly based on inductive logic programming (Zelle & Mooney, 1996) or deterministic parsing (Kate et al., 2005), which lack some of the robustness that characterizes the learning methods recently developed in statistical NLP. Other work involves no learning at all (Androutsopoulos et al., 1995; Popescu et al., 2003), and hence requires extensive human efforts when porting it to a new application domain.

In this proposal, we present a novel statistical approach to semantic parsing which can handle MRs with a nested structure, based on our previous work on semantic parsing using transformation rules (Kate et al., 2005). The algorithm learns a semantic parser given a set of NL sentences annotated with their correct MRs. It requires no prior knowledge of the NL syntax, although it assumes that an unambiguous, context-free grammar (CFG) of the target meaning-representation language (MRL) is available. The main innovation of this algorithm is its integration with state-of-the-art statistical machine translation techniques. More specifically, a statistical word alignment model (Brown et al., 1993) is used for acquiring a bilingual lexicon consisting of NL substrings coupled with their translations in the target MRL. Complete MRs are then formed by combining these NL substrings and their translations under a synchronous parsing framework called the synchronous CFG (Aho & Ullman, 1972), which forms the basis of most existing statistical syntax-based translation models (Yamada & Knight, 2001; Chiang, 2005). In initial evaluation on several real-world data sets, we show that this algorithm performs favorably in terms of both accuracy and coverage compared to existing learning methods requiring the same amount of supervision, and shows better robustness to variations in task complexity and word order.

In future work, we will pursue several directions in developing better semantic parsing algorithms for a wider variety of application domains:

1. Extending the current semantic parsing algorithm to exploit any syntactic information of input sentences that is made available, as has been done for machine translation (Och et al., 2003) and semantic role labeling (Carreras & Màrquez, 2005);

2. Utilizing additional knowledge about the application domain to help disambiguate semantic parses, by choosing semantic interpretations that are more plausible (Schuler, 2003);

3. Constructing and developing word-based alignment models that are aware of the MRL syntax for more effective lexical acquisition;
4. Extending the algorithm to handle linguistic phenomena such as anaphora and discourse;
5. Generalizing the learning algorithm to accept noisy training data, which is a prerequisite to language learning from multi-modal sensory input (André, 2003; Gorniak & Roy, 2004).

The remainder of this proposal is organized as follows. Section 2 reviews several recently-developed semantic parsing algorithms and their application domains, as well as prior work on synchronous parsing and statistical machine translation. In Section 3, we describe our new semantic parsing algorithm, WASP, and present some initial experimental results. In Section 4, we outline our proposed future research directions.

2 Background and Related Work

2.1 Semantic Parsing

Since the last decade, a number of algorithms have been developed for learning semantic parsers that map an NL to an MRL. Given a training corpus of NL sentences annotated with their correct semantic interpretation in a given MRL, the goal of these algorithms is to induce an accurate semantic parser that can map novel sentences into the target MRL. In this section, we first describe the applications that have been explored, and provide a brief overview of the semantic parsing algorithms that have been tested on these applications.

2.1.1 Application Domains

Previous research in semantic parsing has mainly focused on the following three domains. The first one is the ATIS domain, an ARPA-sponsored benchmark for speech recognition and understanding (Price, 1990). The ATIS corpus consists of spoken questions about air travel, their written form, and their correct translations in the SQL database query language. The recorded subjects were engaged in a dialog with the database system. Hence the sentences may be semantically dependent on earlier discourse (e.g. in the form of follow-up questions), or may be semantically independent (e.g. the first question in a dialog). Existing semantic parsers typically handle the context-dependent sentences using a hand-coded discourse module (Kuhn & De Mori, 1995), a probabilistic discourse model (Miller et al., 1996), or ignore them altogether (He & Young, 2003; Popescu et al., 2004). The questions are relatively simple, and the task of understanding the questions can be boiled down to slot filling of a single semantic frame (Kuhn & De Mori, 1995; Miller et al., 1996). Below is a sample SQL query with its English gloss:

```
SELECT flight_id FROM flight WHERE from_airport = 'boston'
    AND to_airport = 'new york'
```
Show me flights from Boston to New York.

The second domain is the GEOQUERY domain, where a logical query language based on Prolog is used for querying a small database on U.S. geography. This database was originally chosen due to the availability of a hand-built natural-language interface, GEOBASE, supplied with Turbo Prolog 2.0 (Borland International, 1988). The query language consists of Prolog queries augmented with several meta-predicates (Zelle & Mooney, 1996). A functional, variable-free version of the query language was later developed by Kate et al. (2005), which can be seen to have a set-theoretic interpretation (for an alternative definition, see Section 4.4.1). For example, in the original query language, `countryid(usa)` is a term that refers to the country of the U.S. In the functional query language, it refers to a singleton set that consists of the country of the U.S. In the original query language, the binary predicate `loc(X,Y)` is true if a place, `X`, is located in another place, `Y`. In the functional query language, `loc_1` is a binary relation that takes a finite set of places, X, and returns a finite set of places that X is located in. The number `1` in the name `loc_1` indicates that X corresponds to the first argument of the binary predicate `loc`. Similarly, `loc_2` is a binary relation that takes a finite set of places, Y, and returns a finite set of places that are located in Y. Queries in this domain typically show a deeply nested structure, in contrast to the flat structure of an ATIS query. Below is a sample Prolog query, its functional, variable-free form, and its English gloss:

```
answer(A,count(B,(city(B),loc(B,C),const(C,countryid(usa))),A))
answer(count(city(loc_2(countryid(usa)))))
```
How many cities are there in the US?

The same Prolog query language has also been used to build NLIs for databases of restaurants and CS-job openings (Tang & Mooney, 2000; Popescu et al., 2003).

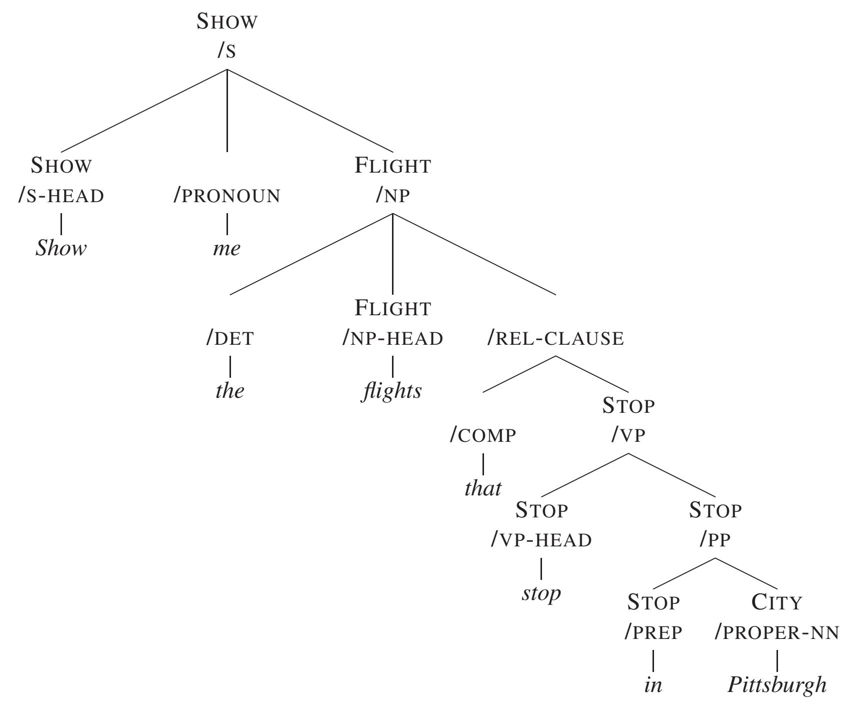
The third domain is the ROBOCUP domain. ROBOCUP (`www.robocup.org`) is an international AI research initiative using robotic soccer as its primary domain. In the ROBOCUP Coach Competition, teams of agents compete on a simulated soccer field and receive advice from a team coach in a formal, functional 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. This domain was used in Kate et al. (2005) and Ge and Mooney (2005) for evaluating semantic parsers. 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 our penalty area, all our players except player 4 should stay in our half.

In the corpora developed for both GEOQUERY and ROBOCUP domains, all sentences are context-independent.

2.1.2 Syntax-Based Approaches

One of the earliest approaches to statistical semantic parsing is based on the idea of extending syntactic parsers with semantic labels. Miller et al. (1994) introduces a tree-structured MR that is structurally equivalent to a full syntactic parse tree. To model meanings, some of the node labels are augmented to reflect semantic categories. Each semantically-labeled node represents an abstract concept, with component concepts appearing as nodes positioned below it. The resulting tree is called an *augmented parse tree*. Figure 1(a) shows a sample augmented parse for the sentence *Show me the flights that stop in Pittsburgh* in the ATIS domain. Here the FLIGHT node represents the abstract concept of a flight, which is a structured entity containing a stopover event, represented by the STOP nodes. The stopover event in turn contains a CITY as its sole component, which is *Pittsburgh* in this case. Note the use of syntactic labels such as NP (noun phrase) and VP (verb phrase). There are also purely syntactic nodes such as DET (determiner) and COMP (complementizer) that do not contribute to the overall meaning of the sentence. Miller et al. (1994) proposes a parsing model based on a probabilistic recursive transition network. The augmented parse tree



(a) An augmented parse tree

$$\begin{bmatrix} \textit{Air-Transportation} & \\ \textsc{show} & \text{Flight} \\ \textsc{stop} & \begin{bmatrix} \textsc{city} & \textit{Pittsburgh} \end{bmatrix} \end{bmatrix}$$

(b) An equivalent semantic frame

Figure 1: Sample MRs in the ATIS domain taken from Miller et al. (1994, 1996)

can be converted into a non-recursive semantic frame (Figure 1(b)) using a probabilistic semantic interpretation model (Miller et al., 1996). Parameters for both models are estimated using fully-annotated augmented parses as the training data.

Ge and Mooney (2005) presents an extension to this semantic parsing framework called SCISSOR. It is an improvement in two respects. First, SCISSOR is based on a state-of-the-art syntactic parsing model, Collins's (1997) Model 2. The parsing model is *lexicalized*, as in most high performance syntactic parsers (Magerman, 1995; Charniak, 2000), where each internal node in an augmented parse is annotated with a head word and the syntactic and semantic labels of the head word's pre-terminal. Second, for recovering an MR from an augmented parse, a simple recursive procedure is devised that allows an MR to be multiple levels deep, which is the case in both ROBOCUP and GEOQUERY domains (as opposed to Miller et al. (1996) where output semantic frames are flat). Again, fully-annotated augmented parses are used for training a parsing model.

The main drawback of both Miller et al. (1994, 1996) and Ge and Mooney (2005) is that they require fully-annotated augmented parses for training, which often mean an excessive amount of work for annotators. Zettlemoyer and Collins (2005) addresses this problem by allowing training sentences to be a simple word sequence. Each training sentence is then coupled with its correct MR, with no extra semantic annotations on each NL phrase. Their method is based on a combinatory categorial grammar (CCG). It requires prior knowledge of syntax, which comes in the form of rules for constructing a bilingual lexicon. Each rule consists of an input trigger and an output category, which we illustrate using an example:

$$\begin{aligned} \textit{Input trigger:}&\ \text{any binary predicate } p \\ \textit{Output category:}&\ (\text{S}\backslash\text{NP})/\text{NP} : \lambda x.\lambda y.p(y,x) \end{aligned}$$

Suppose we are given a training sentence, *Utah borders Idaho*, and its logical form, $\textsc{borders}(\textsc{utah}, \textsc{idaho})$. Since the logical form contains a binary predicate BORDERS, the above rule is *triggered*, producing a lexical item for each word in the training sentence:

$$\begin{aligned} \textit{Utah} &= (\text{S}\backslash\text{NP})/\text{NP} : \lambda x.\lambda y.\textsc{borders}(y,x) \\ \textit{borders} &= (\text{S}\backslash\text{NP})/\text{NP} : \lambda x.\lambda y.\textsc{borders}(y,x) \\ \textit{Idaho} &= (\text{S}\backslash\text{NP})/\text{NP} : \lambda x.\lambda y.\textsc{borders}(y,x) \end{aligned}$$

These lexical items specify that each word is of the syntactic category $(\text{S}\backslash\text{NP})/\text{NP}$, which takes an NP (noun phrase) to the right (/NP), and then an NP to the left (\NP), forming an S (sentence) constituent. The logical forms associated with the NPs combine with the lambda function $\lambda x.\lambda y.\textsc{borders}(y,x)$ through function applications, giving rise to a grounded formula. Similar lexicon-generating rules are devised for constants (e.g. UTAH and IDAHO) and other types of predicates and functions. The goal is to generate a set of lexical items that are sufficient for parsing a training sentence correctly. Note that these rules are specific to a particular NL and MRL pair. A change in the NL syntax or the application domain would require a new set of lexicon-generating rules, although it should be easier for annotators to come up with a set of rules than to produce a detailed augmented parse tree for each training sentence. Also note that the lexicon contains many spurious lexical items, such as the one that associates *Utah* with BORDERS. These spurious items are pruned from the lexicon during the parameter estimation phase, where the weights of lexical items are estimated according to the maximum-entropy principle in an unsupervised fashion (since fully-annotated augmented parses are not available for training).

A major innovation of these syntax-based methods is the combination of syntactic parsing and semantic parsing as a single process. Since ambiguities arise in both the syntactic structure and the semantic interpretation of phrases, the combination of syntactic and semantic parsing as a single model allows semantic information to be used to resolve syntactic ambiguities, and syntactic information to be used to resolve semantic ambiguities.

2.1.3 Semantic Grammars

One thing that syntax-based methods share in common is that a *full* syntactic parse is generated alongside a semantic parse. Thus, a parser often constructs a more elaborate structure than needed to recover an MR (e.g. Figure 1). One way to simplify the output structure is to remove syntactic labels and purely-syntactic nodes, such that the induction of a parser is entirely driven by the semantic representations of training sentences. This results in a *semantic grammar* (Allen, 1995), where non-terminals correspond to concepts in an application domain as opposed to syntactic categories. Figure 2 shows a possible purely-semantic parse equivalent to the one in Figure 1.

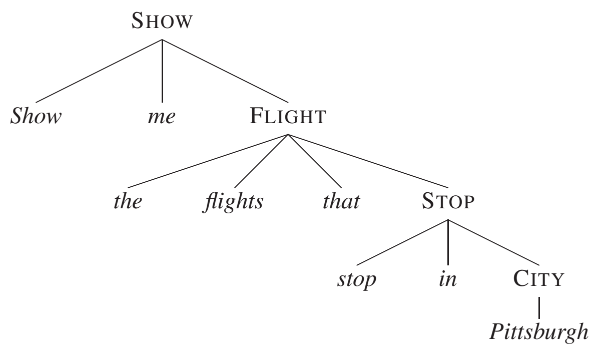


Figure 2: A sample semantic parse in the ATIS domain

Kate et al. (2005) presents an algorithm called SILT for inducing a semantic grammar. It is based on *transformation rules* that map NL substrings to MRs. For example, a transformation rule can be formed that maps the word *Pittsburgh* to an MR $\left[\text{CITY } \textit{Pittsburgh}\right]$. Another transformation rule can be formed that maps the substring *stop in* x to a slot-value pair $\text{STOP} = x'$, where x is any city name and x' is the MR representing the city. To transform the substring *stop in Pittsburgh* into an MR, the substring is parsed using these transformation rules, giving the semantic parse in Figure 2. At the same time, an MR is generated by composing $\text{STOP} = x'$ with $\left[\text{CITY } \textit{Pittsburgh}\right]$, giving a grounded slot-value pair $\text{STOP} = \left[\text{CITY } \textit{Pittsburgh}\right]$. Such transformation rules are learned in a bottom-up manner, starting from constants such as $\left[\text{CITY } \textit{Pittsburgh}\right]$, then structured entities such as $\text{STOP} = x'$, up to the top-level concepts, such as the creation of *Air-Transportation* frames. Fully-annotated semantic parses are not needed for training a semantic parser, although it does require prior knowledge of the grammar of the target MRL.

2.1.4 Other Approaches

Various other learning approaches have been proposed for semantic parsing. Zelle and Mooney (1996) presents a system called CHILL which uses induction logic programming (ILP) (Muggleton, 1992) to learn a deterministic shift-reduce parser written in Prolog. CHILL treats parser induction as a problem of learning rules to control the actions of a shift-reduce parser. There are three types of operators. First is the introduction of a predicate due to the appearance of a word. Second is the unification of variables that have been previously introduced. Third is the embedding of a predicate as an argument of a previously-introduced meta-predicate. The learning algorithm requires a semantic lexicon that provides the possible mapping from words to predicates. Such a lexicon can be automatically acquired from a set of training sentences and their correct MRs (Thompson & Mooney, 1999). Tang and Mooney (2001) presents an extension to CHILL called COCKTAIL which uses multiple clause constructors. The resulting parser is shown to have better coverage compared to the original CHILL algorithm.

He and Young (2003) introduces a semantic parser based on an extended hidden Markov model (HMM) (Rabiner, 1989) that allows hierarchical structures to be efficiently represented while retaining the computational tractability of regular HMMs. Given a semantic parse as shown in Figure 2, the semantic information related to any single word is stored as a vector of semantic labels starting from the pre-terminal of the word, and ending at the root node. For example, the word *stop* would be described by the vector $\langle \text{STOP}, \text{FLIGHT}, \text{SHOW} \rangle$ (cf. Figure 2). Hence a complete semantic parse can be represented by a sequence of vectors. Inference and induction follow the standard HMM algorithms (Jelinek, 1998). This method is shown to be effective for parsing context-independent sentences in the ATIS domain.

Papineni et al. (1997) and Macherey et al. (2001) are two semantic parsing systems based on machine translation algorithms, which will be introduced in Section 2.3. Both algorithms translate English questions directly into SQL statements as if the target MRs consist of strings of tokens. Papineni et al. (1997) is based on a discriminatively-trained, word-based translation model (Section 2.3.1), while Macherey et al. (2001) is based on a phrase-based translation model (Section 2.3.2). Like He and Young (2003), these algorithms are designed to handle simple SQL queries in the ATIS domain.

Kuhn and De Mori (1995) introduces a system called CHANEL that translates NL questions into SQL queries based on classifications given by decision trees. Each decision tree decides whether to include a particular displayed attribute or constraint in the output SQL query. Decisions are based on pattern matching of the input. For example, a matching of the pattern *"stop in"* would indicate the presence of a STOP constraint in the query. Decision trees are learned from a set of training sentences and their corresponding SQL queries. Like Miller et al. (1996), CHANEL is designed to handle simple SQL queries that are non-recursive.

PRECISE (Popescu et al., 2003, 2004) is a recent approach to constructing an NL interface to databases which does *not* involve any learning. It introduces the notion of *semantically tractable* sentences, sentences that give rise to a unique semantic interpretation given a partly manually-constructed lexicon and a set of semantic constraints. The lexicon specifies the semantic types of a word. For example, by looking up the lexicon, the word *in* is found to be an attribute token, and *Pittsburgh* a value token. Each semantic constraint specifies a restriction on the types of values that an attribute can take. For example, the attribute IN can only take a city or an airport as its value. By mapping each word in an input sentence to a set of possible lexical items, semantic parsing is reduced to the problem of maximum flow in a graph that connects attributes and values together according to the semantic constraints. Interestingly, Popescu et al. (2004) shows that over 90% of the context-independent ATIS questions are semantically tractable (i.e. for which correct mapping to an SQL query is guaranteed). On the other hand, only 80% of the GEOQUERY questions are semantically tractable, which shows that GEOQUERY is indeed a more challenging domain than ATIS.

2.2 Synchronous Parsing

In this section, we present some notions that formalize the process of semantic parsing. We first explore semantic parsing from an abstract point of view, and then consider how the resulting parsing model relates to semantic grammars (Section 2.1.3).

Semantic parsing can be shown as an instance of *syntax-directed translations* (Aho & Ullman, 1969a, 1969b, 1972), originally developed as a theory of compilers in which syntax analysis and code generation are combined into a single phase (i.e. such that a high-level, formal programming language is translated into machine code, another formal language). The theory is also known as *synchronous parsing*, a term more commonly used in the NLP community (Wu, 1997; Melamed, 2004). According to this theory, a semantic parser defines a *translation*, a set of pairs of strings in which each pair is an NL sentence coupled with its MR. Suppose that $\mathcal{T}_n$ is an NL alphabet and $\mathcal{T}_m$ is an MRL alphabet. We define a translation from an NL $L_n \subseteq \mathcal{T}_n^*$ to an MRL $L_m \subseteq \mathcal{T}_m^*$ as a relation from $\mathcal{T}_n^*$ to $\mathcal{T}_m^*$. For example, in the ROBOCUP domain, $\mathcal{T}_m$ would be the set $\{\texttt{true}, \texttt{do}, \ldots\}$, and a translation from English into CLANG would contain the following pair: *Our player 4 should always stay in our half*, `((true) (do our {4} (pos (half our))))`.

There are several possible approaches to the finite specification of a potentially infinite translation. Analogous to an ordinary grammar, we can have a system which generates the pairs in a translation. Or we can define a parser which takes a string x as input and produces (perhaps non-deterministically) all y such that $\langle x, y \rangle$ is in a translation (or y is a *translation of* x). While the latter is a more common way to look at a semantic parser, we note that these two views are indeed identical. One formalism for defining translations is the *synchronous context-free grammar* (or the *syntax-directed translation schema*). A synchronous context-free grammar (SCFG) is a 5-tuple:

$$G = \langle \mathcal{N}, \mathcal{T}_n, \mathcal{T}_m, \mathcal{L}, S \rangle \tag{1}$$

where $\mathcal{N}$ is a finite set of non-terminal symbols, T_n is a finite set of words (terminals) of the NL, T_m is a finite set of terminals of the MRL, S is a non-terminal in $\mathcal{N}$ called the *start symbol*, and $\mathcal{L}$ is a finite set of rules of the form $X \to \langle \alpha, \beta \rangle$, where $\alpha \in (\mathcal{N} \cup \mathcal{T}_n)^*$, $\beta \in (\mathcal{N} \cup \mathcal{T}_m)^*$, and the non-terminals in β are a permutation of the non-terminals in α.

Suppose that $X \to \langle \alpha, \beta \rangle$ is a rule. To each non-terminal in α there is an *associated*, identical non-terminal in β. If each non-terminal appears only once in α or β, then the association is obvious. Otherwise, indices are needed to indicate the association. For example, in the rule $X \to \langle Y_{\boxed{1}} Y_{\boxed{2}}, Y_{\boxed{2}} Y_{\boxed{1}} \rangle$, the two positions in $Y_{\boxed{1}} Y_{\boxed{2}}$ are associated with positions 2 and 1 in $Y_{\boxed{2}} Y_{\boxed{1}}$, respectively. In this proposal, indices are explicitly written in all cases. We define a *translation form* of G as follows:

1. $\langle S_{\boxed{1}}, S_{\boxed{1}} \rangle$ is a translation form.
2. If $\langle \alpha X_{\boxed{i}} \beta, \alpha' X_{\boxed{i}} \beta' \rangle$ is a translation form, and if $X \to \langle \gamma, \gamma' \rangle$ is a rule in $\mathcal{L}$, then $\langle \alpha\gamma\beta, \alpha'\gamma'\beta' \rangle$ is a translation form. The non-terminals in γ and γ' are associated in the translation form exactly as they are associated in the rule. The non-terminals of α and β are associated with those of α' and β' in the new translation form exactly as in the old.

If the forms $\langle \alpha X_{\boxed{i}} \beta, \alpha' X_{\boxed{i}} \beta' \rangle$ and $\langle \alpha\gamma\beta, \alpha'\gamma'\beta' \rangle$ are related as above, then we write $\langle \alpha X_{\boxed{i}} \beta, \alpha' X_{\boxed{i}} \beta' \rangle \Rightarrow_G \langle \alpha\gamma\beta, \alpha'\gamma'\beta' \rangle$. The non-terminals $X_{\boxed{i}}$ in $\langle \alpha X_{\boxed{i}} \beta, \alpha' X_{\boxed{i}} \beta' \rangle$ are said to be *rewritten* by the rule $X \to \langle \gamma, \gamma' \rangle$, and we say the rule has been *applied*. The *translation defined by* G, denoted $\tau(G)$, is the set of pairs:

$$\{\langle x, y \rangle | \langle S_{\boxed{1}}, S_{\boxed{1}} \rangle \Rightarrow_G^* \langle x, y \rangle, x \in \mathcal{T}_n^*, y \in \mathcal{T}_m^*\} \tag{2}$$

where $\Rightarrow_G^*$ is the reflexive-transitive closure of $\Rightarrow_G$.

We further define the *input grammar* of G as the 4-tuple $G_n = \langle \mathcal{N}, \mathcal{T}_n, \mathcal{L}_n, S \rangle$, where $\mathcal{L}_n = \{X \to \alpha | X \to \langle \alpha, \beta \rangle \in \mathcal{L}\}$. Similarly, the *output grammar* of G is defined as the 4-tuple $G_m = \langle \mathcal{N}, \mathcal{T}_m, \mathcal{L}_m, S \rangle$, where $\mathcal{L}_m = \{X \to \beta | X \to \langle \alpha, \beta \rangle \in \mathcal{L}\}$. Both G_n and G_m are CFGs. We can then view synchronous parsing as a process in which two parse trees are generated simultaneously, one based on the input grammar, the other one based on the output grammar. Furthermore, these two parse trees are *isomorphic* (i.e. there exists a one-to-one mapping from the nodes of one parse tree to another such that the two trees are identical), since the pair of non-terminals rewritten at each step of a derivation are always associated with each other. Alternatively, given an input string x, a semantic parser finds (if possible) some derivation of x from S using the productions in the input grammar. Let $S = \alpha_0 \Rightarrow_{G_n} \alpha_1 \Rightarrow_{G_n} \ldots \Rightarrow_{G_n} \alpha_k = x$ be such a derivation. This corresponds to the following derivation of translation forms of G: $\langle \alpha_0, \beta_0 \rangle \Rightarrow_G \ldots \Rightarrow_G \langle \alpha_k, \beta_k \rangle$, such that $\langle \alpha_0, \beta_0 \rangle = \langle S_{\boxed{1}}, S_{\boxed{1}} \rangle$ and $\langle \alpha_k, \beta_k \rangle = \langle x, y \rangle$. The string y is then a translation of x.

We are now in the position of explaining the relationship between synchronous grammars and semantic grammars. A semantic grammar as described in Section 2.1.3 can be seen as a synchronous grammar where the set of non-terminals $\mathcal{N}$ correspond to abstract concepts in the application domain (e.g. CONDITION and ACTION in the ROBOCUP domain), with a start symbol that corresponds to the top-level concept (e.g. RULE for ROBOCUP). Synchronous grammar is an ideal formalism for defining a semantic grammar because it describes the hierarchical structure of a sentence (e.g. Figure 2), and allows the sentence to be translated into an MR during the same parsing process. In particular, the SILT algorithm (Kate et al., 2005) presented in Section 2.1.3 can be described as an SCFG, where each transformation rule corresponds to an SCFG rule. This is best illustrated using an example. Suppose that we are given an input sentence:

If our player 4 has the ball , our player 4 should shoot .

The task is to translate it into an MR in the ROBOCUP domain. This can be done by an SCFG, $G = \langle \mathcal{N}, \mathcal{T}_n, \mathcal{T}_m, \mathcal{L}, S \rangle$, where:

$$\begin{aligned}
\mathcal{N} &= \{\text{TEAM}, \text{UNUM}, \text{ACTION}, \text{CONDITION}, \text{DIRECTIVE}, \text{RULE}\} \\
\mathcal{T}_n &= \{\textit{if}, \textit{our}, \textit{player}, \textit{4}, \ldots\} \\
\mathcal{T}_m &= \{\,\texttt{(}, \texttt{bowner}, \texttt{our}, \texttt{4}, \ldots\} \\
\mathcal{L} &= \{\text{RULE} \to \langle\, \textit{if}\ \text{CONDITION}_{\boxed{1}}\ (1)\ \text{DIRECTIVE}_{\boxed{2}}\ (1)\ ,\ \texttt{(}\text{CONDITION}_{\boxed{1}}\ \text{DIRECTIVE}_{\boxed{2}}\texttt{)}\ \rangle, \\
&\qquad \text{CONDITION} \to \langle\, \text{TEAM}_{\boxed{1}}\ \textit{player}\ \text{UNUM}_{\boxed{2}}\ \textit{has}\ (1)\ \textit{ball}\ ,\ \texttt{(bowner}\ \text{TEAM}_{\boxed{1}}\ \texttt{\{}\text{UNUM}_{\boxed{2}}\texttt{\})}\ \rangle, \\
&\qquad \text{TEAM} \to \langle\, \textit{our}\ ,\ \texttt{our}\ \rangle, \\
&\qquad \text{UNUM} \to \langle\, \textit{4}\ ,\ \texttt{4}\ \rangle, \\
&\qquad \text{DIRECTIVE} \to \langle\, \text{TEAM}_{\boxed{1}}\ \textit{player}\ \text{UNUM}_{\boxed{2}}\ \textit{should}\ \text{ACTION}_{\boxed{3}}\ ,\ \texttt{(do}\ \text{TEAM}_{\boxed{1}}\texttt{\{}\text{UNUM}_{\boxed{2}}\texttt{\}}\ \text{ACTION}_{\boxed{3}}\texttt{)}\ \rangle, \\
&\qquad \text{ACTION} \to \langle\, \textit{shoot}\ ,\ \texttt{(shoot)}\ \rangle\} \\
S &= \text{RULE}
\end{aligned}$$

Here each rule in $\mathcal{L}$ is a SILT transformation rule. The special (1) symbol is a *word gap*, which indicates that at most one word can be skipped over during pattern matching of the input sentence. For now, it can be seen as a terminal symbol in $\mathcal{T}_n$ which can match at most one word in the input sentence (a more precise description will be given in Section 3.3.1). By parsing the input sentence using the input grammar of G, we obtain the following derivation, where each translation form is a pair of NL and MR strings. This derivation is said to be *top-down*, because it starts with an associated pair of start symbols. It is said to be *left-most*, because at each step of the derivation, the non-terminal rewritten on the NL side is always the left-most one:

$$\langle\, \text{RULE}_{\boxed{1}}\ ,\ \text{RULE}_{\boxed{1}}\ \rangle$$

$$\Rightarrow_G \langle\, \textit{if}\ \text{CONDITION}_{\boxed{1}}\ (1)\ \text{DIRECTIVE}_{\boxed{2}}\ (1)\ ,\ \texttt{(}\text{CONDITION}_{\boxed{1}}\ \text{DIRECTIVE}_{\boxed{2}}\texttt{)}\ \rangle$$

(by applying the 1st rule in $\mathcal{L}$)

$$\Rightarrow_G \langle\, \textit{if}\ \text{TEAM}_{\boxed{1}}\ \textit{player}\ \text{UNUM}_{\boxed{2}}\ \textit{has}\ (1)\ \textit{ball}\ (1)\ \text{DIRECTIVE}_{\boxed{3}}\ (1)\ , \\ \texttt{((bowner}\ \text{TEAM}_{\boxed{1}}\ \texttt{\{}\text{UNUM}_{\boxed{2}}\texttt{\})}\ \text{DIRECTIVE}_{\boxed{3}}\texttt{)}\ \rangle$$

(by applying the 2nd rule in $\mathcal{L}$)

$$\Rightarrow_G \langle\, \textit{if our player}\ \text{UNUM}_{\boxed{1}}\ \textit{has}\ (1)\ \textit{ball}\ (1)\ \text{DIRECTIVE}_{\boxed{2}}\ (1)\ , \\ \texttt{((bowner our \{}\text{UNUM}_{\boxed{1}}\texttt{\})}\ \text{DIRECTIVE}_{\boxed{2}}\texttt{)}\ \rangle$$

(by applying the 3rd rule in $\mathcal{L}$)

$$\Rightarrow_G \langle\, \textit{if our player 4 has}\ (1)\ \textit{ball}\ (1)\ \text{DIRECTIVE}_{\boxed{1}}\ (1)\ , \\ \texttt{((bowner our \{4\})}\ \text{DIRECTIVE}_{\boxed{2}}\texttt{)}\ \rangle$$

(by applying the 4th rule in $\mathcal{L}$)

$$\Rightarrow_G \langle\, \textit{if our player 4 has}\ (1)\ \textit{ball}\ (1)\ \text{TEAM}_{\boxed{1}}\ \textit{player}\ \text{UNUM}_{\boxed{2}}\ \textit{should}\ \text{ACTION}_{\boxed{3}}\ (1)\ , \\ \texttt{((bowner our \{4\}) (do}\ \text{TEAM}_{\boxed{1}}\ \texttt{\{}\text{TEAM}_{\boxed{2}}\texttt{\}}\ \text{ACTION}_{\boxed{3}}\texttt{))}\ \rangle$$

(by applying the 5th rule in $\mathcal{L}$)

$$\Rightarrow_G \langle\, \textit{if our player 4 has}\ (1)\ \textit{ball}\ (1)\ \textit{our player}\ \text{UNUM}_{\boxed{1}}\ \textit{should}\ \text{ACTION}_{\boxed{2}}\ (1)\ , \\ \texttt{((bowner our \{4\}) (do our \{}\text{TEAM}_{\boxed{1}}\texttt{\}}\ \text{ACTION}_{\boxed{2}}\texttt{))}\ \rangle$$

(by applying the 3rd rule in $\mathcal{L}$)

$\Rightarrow_G \langle$ *if our player 4 has* (1) *ball* (1) *our player 4 should* ACTION$_{\boxed{1}}$ (1) ,
`((bowner our {4}) (do our {4}` ACTION$_{\boxed{1}}$`))` $\rangle$

(by applying the 4th rule in $\mathcal{L}$)

$\Rightarrow_G \langle$ *if our player 4 has* (1) *ball* (1) *our player 4 should shoot* (1) ,
`((bowner our {4}) (do our {4} (shoot)))` $\rangle$

(by applying the 6th rule in $\mathcal{L}$)

Each step of this derivation corresponds to an application of a SILT transformation rule. This derivation yields an NL string, *if our player 4 has* (1) *ball* (1) *our player 4 should shoot* (1), and an MR string, `((bowner our {4}) (do our {4} (shoot)))`. Since the NL string matches the input sentence by skipping over the word *the*, the comma, and the period, the MR string is a translation of the input sentence.

In general, an input sentence, x, is parsed using the productions in the input grammar of G. Each parse corresponds to a top-down derivation of translation forms of G. The MR string in the final translation form in this derivation is then a possible translation of x. Since each parse of x corresponds to a semantic parse tree (e.g. Figure 2), G can be seen as a semantic grammar that translates an NL sentence into an MR string.

Apart from SILT, Shieber and Schabes (1990) presents another system that applies synchronous parsing to natural languages. It introduces the *synchronous tree-adjoining grammar* (STAG) as a formalism for semantic parsing. STAG is similar to SCFG except that its underlying grammar is a tree-adjoining grammar (TAG) (Joshi, 1985).

2.3 Statistical Machine Translation

Another line of work that will be relevant to the task of semantic parsing is machine translation, whose main goal is to translate one natural language into another. Although this task is similar to semantic parsing, where a natural language is translated into a formal language, and syntax-directed translation, where a formal language is translated into another, machine translation often presents unique challenges of its own, because of the inherent ambiguity of natural languages. Machine translation is thus a particularly challenging task, which has inspired a large body of research. In particular, the growing availability of bilingual corpora, where the same content is available in two languages, has stimulated interest in statistical methods for extracting linguistically valuable information from such texts. In this section, we will review the main components of a typical statistical machine translation system. Then we will focus on the sub-task of *word alignment*, which will be the most relevant to our work.

Without loss of generality, machine translation is the task of translating a string of foreign words, $\mathbf{f}$, into a string of English words, $\mathbf{e}$. Obviously, there are many acceptable translations for a given $\mathbf{f}$. The choice would depend on the context in which $\mathbf{f}$ appears, the level of the target audience, and so on. In statistical machine translation, we take the view that every English sentence, $\mathbf{e}$, is a possible translation of $\mathbf{f}$. Every pair of strings $\langle \mathbf{e}, \mathbf{f} \rangle$ is assigned a probability $\Pr(\mathbf{e}|\mathbf{f})$. The task of translating the foreign string $\mathbf{f}$ is then to choose the English string $\mathbf{e}^\star$ for which $\Pr(\mathbf{e}|\mathbf{f})$ is the greatest. Traditionally, this task is divided into two separate, more manageable sub-tasks:

$$\mathbf{e}^\star = \arg\max_{\mathbf{e}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e}) \tag{3}$$

This decomposition is based on the noisy-channel paradigm. The induction of a machine translation system thus involves the estimation of two probability distributions, namely the *language model* (or *source model*),

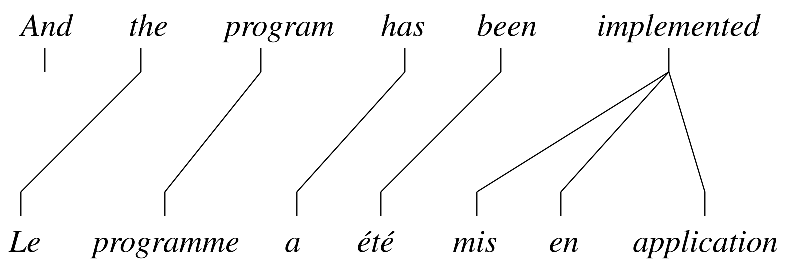


Figure 3: A sample word alignment, taken from Brown et al. (1993)

$\Pr(\mathbf{e})$, and the *translation model* (or *channel model*), $\Pr(\mathbf{f}|\mathbf{e})$. These two models cooperate to produce English translations that are well-formed and explain the foreign string well.

The latter model is also known as the *alignment model*, because given a *pair* of strings, $\mathbf{e}$ and $\mathbf{f}$, the model explains how words in $\mathbf{e}$ are translated into words in $\mathbf{f}$. Figure 3 shows a sample *word alignment* between an English string and a French string. It shows that the French word *le* is a translation of (or is *linked to*) the English word *the*, the French phrase *mis en application* as a whole is a translation of the English word *implemented*, and so on. Numerous statistical methods have been devised to find the best word alignment between sentences in a bilingual corpora, and it is in this sense that machine translation becomes relevant to the task of semantic parsing: a word alignment defines a bilingual lexicon. In Figure 3, the meanings of the French words and phrases are expressed in English. The semantic parsing scenario is similar, except that the meanings are expressed in a formal representation.

Therefore, alignment models will be the main focus of our discussion. In the following text, we will review some early word-based alignment models, and then go on to discuss phrase-based alignment models and their generalizations.

2.3.1 Word-Based Alignment Models

Brown et al. (1993) presents a series of five statistical translation models which later became known as the *IBM Models*. These models are *word-based* because they model how each individual word in $\mathbf{e}$ is translated into words in $\mathbf{f}$, resulting in a 1-to-n alignment (Figure 3). Suppose that $\mathbf{f} = f_1^J = \langle f_1, \dots, f_J \rangle$, and $\mathbf{e} = e_1^I = \langle e_1, \dots, e_I \rangle$. A word alignment, $\mathbf{a}$, between $\mathbf{f}$ and $\mathbf{e}$ is defined as:

$$\mathbf{a} = \langle a_1, \dots, a_J \rangle, \forall j = 1, \dots, J, 0 \le a_j \le I \tag{4}$$

where a_j is the position of the English word that the foreign word f_j is linked to. If $a_j = 0$, then f_j is not linked to any English word. The conditional probability $\Pr(\mathbf{f}|\mathbf{e})$ can then be written as follows:

$$\Pr(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} \Pr(\mathbf{f}, \mathbf{a}|\mathbf{e}) \tag{5}$$

Each IBM model gives a prescription for decomposing the conditional probability $\Pr(\mathbf{f}, \mathbf{a}|\mathbf{e})$ such that inference and induction are tractable. For Models 1 and 2, the conditional probability is decomposed as follows:

$$\Pr(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \Pr(J|\mathbf{e}) \prod_{j=1}^{J} \Pr(a_j|a_1^{j-1}, f_1^{j-1}, J, \mathbf{e}) \Pr(f_j|a_1^j, f_1^{j-1}, J, \mathbf{e}) \tag{6}$$

Using this decomposition, we obtain three different probabilities, namely a length probability $\Pr(J|\mathbf{e})$, an alignment probability $\Pr(a_j|a_1^{j-1}, f_1^{j-1}, J, \mathbf{e})$, and a translation probability $\Pr(f_j|a_1^j, f_1^{j-1}, J, \mathbf{e})$. We further assume that the length probability is independent of J and $\mathbf{e}$; that the alignment probability depends only on a_j, j, I and J; and that the translation probability depends only on f_j and e_{a_j}. We can therefore write the length probability as a constant ϵ, the alignment probability as $\Pr(a_j|j, I, J)$, and the translation probability as $\Pr(f_j|e_{a_j})$. Models 1 and 2 are said to be *zero-order* because a_j is independent of a_1^{j-1} (i.e. the movement of each word is independent). Furthermore, Model 1 assumes a uniform distribution for the alignment probability $\Pr(a_j|j, I, J) = 1/(I + 1)$. Hence the word order in either string does not affect the alignment probability.

Both Models 1 and 2 are estimated using the Expectation-Maximization (EM) algorithm (Dempster et al., 1977), based on a training set of English sentences coupled with their correct foreign translations. The model parameters are first guessed, and then repeatedly updated such that the likelihood of the training set is maximized. Models 1 and 2 have an especially simple mathematical form such that iterations of the EM algorithm can be computed exactly. In addition, Model 1 has a unique local maximum so that the parameter estimates do not depend on the initial guess. Although Models 1 and 2 often lead to unsatisfactory word alignments due to the zero-order assumption, they are often used for deriving the initial parameters for subsequent models, for which parameter estimation is much more involved.

In Models 3, 4 and 5, a foreign string is constructed by choosing for each word e_i in the English string, first the number of foreign words that will be linked to e_i (or the *fertility* of e_i), then the identity of these foreign words, and finally the actual positions in the foreign string that these words will occupy. Due to their complexity, we do not discuss them in detail here. There is an important point to note, however. In both Models 4 and 5, the position of every foreign word depends on the position of the previous foreign word that is linked to the same English word. Moreover, given a particular English word, e_i, the position of the first foreign word linked to e_i depends on the average position of all foreign words linked to e_{i-1}. Hence these models are said to be *first-order*, as the movements of words are no longer independent. This allows for better modeling of phrasal movements, such that phrases in English can be translated as a unit into the foreign language. The tendency for phrasal translations to stay close together is called *phrasal coherence*, a topic that we will revisit in Sections 3.2.2 and 4.3.

2.3.2 Phrase-Based and Syntax-Based Alignment Models

A major problem with the IBM Models is their lack of linguistic knowledge. For example, syntactic analysis would allow an alignment model to directly handle phrasal reordering, instead of simulating phrasal movements using the first-order assumption. The utility of the IBM models as a translation model is only demonstrated for a structurally similar language pair, English and French (Brown et al., 1993), and the performance of the models is expected to degrade when the language pair has very different word order (e.g. English and Arabic).

One approach to this problem is to introduce the concept of phrases in an alignment model. A basic phrase-based model decomposes the translation from $\mathbf{e}$ to $\mathbf{f}$ into the following steps. First, $\mathbf{e}$ is segmented into phrases $\tilde{e}_1, \ldots, \tilde{e}_K$. Then the phrases $\tilde{e}_k$ are reordered according to some distortion model. Finally, each $\tilde{e}_k$ is translated into a foreign phrase $\tilde{f}$ according to a phrase translation model $\Pr(\tilde{f}|\tilde{e})$. Och et al. (1999) presents the alignment template approach in which a phrasal lexicon, $\{\langle\tilde{e}, \tilde{f}\rangle\}$, is extracted from word alignments obtained in the first stage of training. These aligned phrase pairs are then generalized to form alignment templates, based on word classes induced from the training data. Tillmann (2003) and Venugopal et al. (2003) also extract a phrasal lexicon from word alignments, but without generalization. Marcu and Wong (2002) learns phrasal translations as part of an EM algorithm in which the joint probability $\Pr(\mathbf{f}, \mathbf{e})$

is estimated.

A common weakness of the phrase-based approaches is that they fail to handle nested syntactic structures. This leads to the development of *syntax-based* alignment models, which can be seen as a generalization of phrase-based methods to the case of hierarchical phrases. Wu (1997) introduces the *inversion transduction grammar* as a tool for bilingual text analysis. It is a version of SCFG where the associated non-terminals in a rule are either in the same order or in reverse order. Chiang (2005) is another SCFG-based alignment model in which the number of associated pairs of non-terminals in each rule is restricted to two. Yamada and Knight (2001, 2002) present a tree-to-string translation model which takes the syntactic parse of an English string, $\mathbf{e}$, performs node reordering, insertions and translations, and outputs a foreign string, $\mathbf{f}$, which is the frontier of the transformed tree. This model can be shown as a regular tree transducer (Knight & Graehl, 2005). Quirk et al. (2005) is another tree-to-string translation model that works on dependency trees.

One thing that these syntax-based alignment models share in common is that they are all based on variants of synchronous grammars. Synchronous grammar is an ideal formalism for defining a syntax-based alignment model because it describes not only the hierarchical structures of a sentence and its translation, but also the exact correspondence between their sub-parts. This leads to the observation that machine translation can be viewed as synchronous parsing (Melamed, 2004). A major reason for adopting this view is that, since synchronous grammars are simply a generalization of grammars originally developed for a single language, machine translation systems can be seen as a generalization of ordinary syntactic parsers. This allows the extensive experience that has been gained in syntactic parsing to be drawn on, expecting that similar techniques are applicable to machine translation as well.

3 Completed Research

In this section we describe a novel approach to semantic parsing based on transformation rules, introduced by Kate et al. (2005). Transformation rules are used for mapping substrings in an NL sentence to MRs, and are learned using an off-the-shelf word alignment model. The learning algorithm assumes that the target MRL has a functional, variable-free form (Section 2.1.1), and that an unambiguous CFG of the target MRL is available. The non-terminal symbols of the MRL grammar provide convenient abstractions that enable the construction of general, effective transformation rules. The learning algorithm requires a set of NL sentences paired with their correct MRs as the training data, with no extra semantic annotations on any sub-parts of a sentence. It stands in marked contrast to Miller et al. (1994, 1996) and Ge and Mooney (2005), where fully-annotated augmented parse trees must be given. Our algorithm requires no prior knowledge of the NL syntax, unlike Zettlemoyer and Collins (2005) where language-dependent rules for building a lexicon must be specified. Our new approach is called WASP, short for *Word Alignment-based Semantic Parsing*. We argue that WASP is robust to variations in task complexity and word order, and performs favorably compared to existing learning algorithms that require similar amount of supervision.

3.1 Overview

WASP is based on transformation rules introduced by the SILT algorithm (Section 2.1.3). Transformation rules are used for mapping NL substrings to MRs. Suppose that $\mathcal{N}$ is a finite set of non-terminal symbols of the MRL grammar, $\mathcal{T}_n$ is a finite set of terminal symbols (words) of the NL, and $\mathcal{T}_m$ is a finite set of terminal symbols of the MRL. Then each transformation rule is of the following form:

$$X \to \langle \mathbf{p}, \mathbf{t} \rangle \tag{7}$$

where $X \in \mathcal{N}$, $\mathbf{p} \in (\mathcal{N} \cup \mathcal{T}_n)^+$, and $\mathbf{t} \in (\mathcal{N} \cup \mathcal{T}_m)^+$. X is called the *left-hand side* (LHS) of the rule, $\mathbf{p}$ is called the *pattern*, and $\mathbf{t}$ is called the *template*. The pair $\langle \mathbf{p}, \mathbf{t} \rangle$ is called the *right-hand side* (RHS) of the rule. Each rule is based on a production of an MRL grammar. We illustrate this using a sample transformation rule in the CLANG domain:

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

Here CONDITION, TEAM and UNUM (uniform number) are non-terminal symbols of the CLANG grammar. CONDITION is the LHS non-terminal. $\langle \textsc{Team}_{\boxed{1}}$ *player* $\textsc{Unum}_{\boxed{2}}$ *has* (1) *ball*, `(bowner` $\textsc{Team}_{\boxed{1}}$ `{`$\textsc{Unum}_{\boxed{2}}$`})` $\rangle$ is the RHS. $\textsc{Team}_{\boxed{1}}$ *player* $\textsc{Unum}_{\boxed{2}}$ *has* (1) *ball* is the pattern, and `(bowner` $\textsc{Team}_{\boxed{1}}$ `{`$\textsc{Unum}_{\boxed{2}}$`})` is the template. This rule is based on the CLANG production CONDITION $\rightarrow$ `(bowner` TEAM `{`UNUM`})`, where `bowner` is the ball owner predicate. Throughout this text, we use the *italic* script for NL words, the SMALL CAPS for non-terminals, the (g) notation to denote a word gap of size g, and the indices $\boxed{1}, \boxed{2}, \ldots$ to indicate the association between non-terminals. We also reserve the term *rules* for transformation rules (and SCFG rules), the term *productions* for productions of an MRL grammar, and the term *non-terminals* for non-terminals of an MRL grammar (as opposed to syntactic categories in an NL grammar).

As noted in Section 2.2, a transformation rule is an SCFG rule. Moreover, a semantic parsing model based on transformation rules is an SCFG, G. Note that the input grammar of G is a semantic grammar for the NL, and the output grammar of G is a grammar for the MRL. In Section 2.2, we defined the relation $\Rightarrow_G$, which corresponds to a step in a derivation in G. For example, an application of the above rule to the translation form:

$$\langle\ \textit{if}\ \textsc{Condition}_{\boxed{1}}\ (1)\ \textsc{Directive}_{\boxed{2}}\ (1)\ ,\ \texttt{(}\textsc{Condition}_{\boxed{1}}\ \textsc{Directive}_{\boxed{2}}\texttt{)}\ \rangle$$

would lead to the following translation form:

$$\begin{array}{l}\langle\ \textit{if}\ \textsc{Team}_{\boxed{1}}\ \textit{player}\ \textsc{Unum}_{\boxed{2}}\ \textit{has}\ (1)\ \textit{ball}\ (1)\ \textsc{Directive}_{\boxed{3}}\ (1)\ , \\ \quad \texttt{((bowner}\ \textsc{Team}_{\boxed{1}}\ \texttt{\{}\textsc{Unum}_{\boxed{2}}\texttt{\})}\ \textsc{Directive}_{\boxed{3}}\texttt{)}\ \rangle\end{array}$$

as defined by the $\Rightarrow_G$ relation. The complete MR of an NL sentence, $\mathbf{e}$, is obtained by applying rules repeatedly, starting from an associated pair of start symbols, $\langle S_{\boxed{1}}, S_{\boxed{1}} \rangle$, where $S \in \mathcal{N}$ (e.g. $S = $ RULE for CLANG). Suppose that $\langle S_{\boxed{1}}, S_{\boxed{1}} \rangle \Rightarrow_G \ldots \Rightarrow_G \langle \alpha, \beta \rangle$ is a (top-down) derivation in G. If α *matches* $\mathbf{e}$ (i.e. by skipping over words in $\mathbf{e}$ as specified by the word gaps in α), then β is a possible translation of $\mathbf{e}$. A sample NL sentence and its MR translation is given in Section 2.2.

Formally, the semantic parsing model of WASP is defined by a 6-tuple, G:

$$G = \langle \mathcal{N}, \mathcal{T}_n, \mathcal{T}_m, \mathcal{L}, S, \theta \rangle \tag{8}$$

where $\mathcal{N}$, $\mathcal{T}_n$, $\mathcal{T}_m$, S are defined as above, $\mathcal{L}$ is a *lexicon* which consists of a finite set of transformation rules, and θ is a set of *model parameters*. This definition of G is an extension to the one in (1), by adding θ to the 5-tuple. The model parameters, θ, are for disambiguating the meaning of a sentence when there are multiple possible derivations for it in G. To discriminate the correct translation from the incorrect ones, we use a *probabilistic model*, parameterized by θ, that takes a possible derivation, $\mathbf{d}$, and returns its likelihood to be correct. Derivations are ranked according to their likelihood to be correct as given by the probabilistic model. The output translation, $\mathbf{f}^\star$, for a sentence, $\mathbf{e}$, is defined as:

$$\mathbf{f}^\star = m \left(\operatorname*{arg\,max}_{\mathbf{d} \in D(G) | n(\mathbf{d}) = \mathbf{e}} \Pr\nolimits_\theta(\mathbf{d}) \right) \tag{9}$$

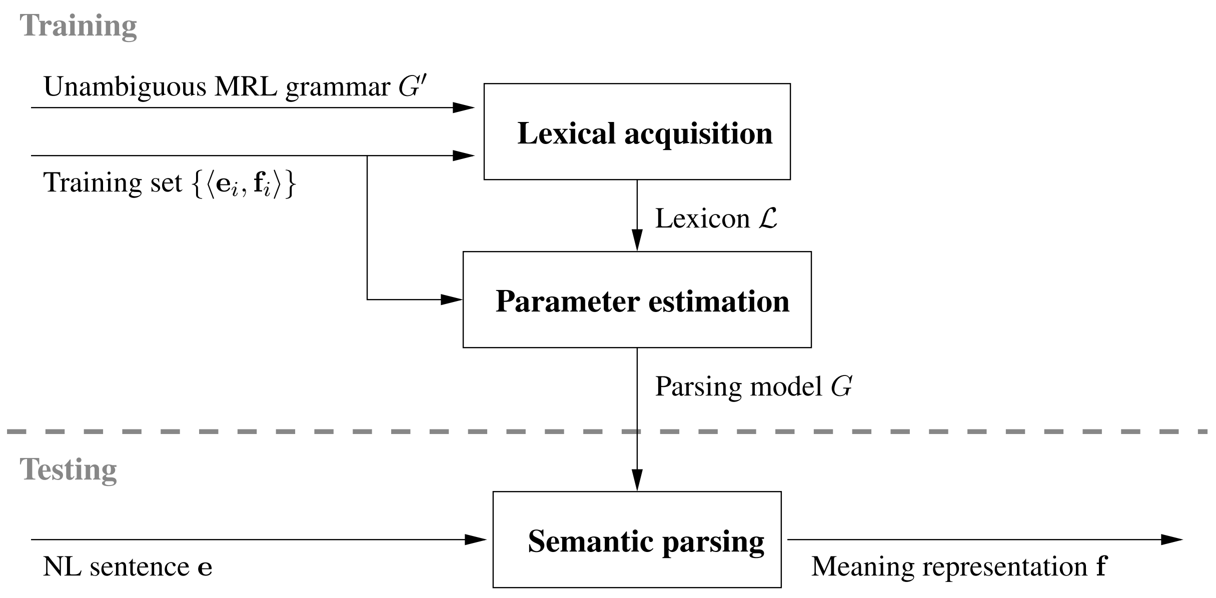


Figure 4: Overview of the proposed semantic parsing framework, WASP

where $m(\mathbf{d})$ is the MR string in the final translation form in $\mathbf{d}$, $n(\mathbf{d})$ is the NL string in the final translation form in $\mathbf{d}$ (by the equal sign in $n(\mathbf{d}) = \mathbf{e}$, we mean that $n(\mathbf{d})$ matches $\mathbf{e}$), and $D(G)$ is the set of all possible derivations given a model G. In other words, the output MR is the yield of the most probable derivation that yields $\mathbf{e}$ in the NL stream. The exact form of $\Pr_\theta(\mathbf{d})$ (and θ) will be given in Section 3.3. This formulation is chosen because $\mathbf{f}^\star$ can be efficiently computed using a dynamic-programming algorithm (Viterbi, 1967).

Since $\mathcal{N}$, T_n, T_m and S are all fixed given an NL and an MRL, the learning task is to induce a lexicon, $\mathcal{L}$, and a probabilistic model, parameterized by θ, from the training data. A lexicon defines the set of derivations that are possible, so the induction of a probabilistic model for derivations requires a lexicon in the first place. Therefore, the learning task can be separated into the following two sub-tasks:

1. Induce a lexicon, $\mathcal{L}$, which implicitly defines the set of all possible derivations, $D(G)$.

2. Induce a probabilistic model, parameterized by θ, which ranks the derivations in $D(G)$.

Both induction tasks require a training set, $\{\langle \mathbf{e}_1, \mathbf{f}_1 \rangle, \langle \mathbf{e}_2, \mathbf{f}_2 \rangle, \ldots\}$, where each training example $\langle \mathbf{e}_i, \mathbf{f}_i \rangle$ is an NL sentence, $\mathbf{e}_i$, paired with its correct MR, $\mathbf{f}_i$. Lexical induction also requires a CFG of the MRL, which is assumed to be unambiguous. Since there is no lexicon to begin with, it is not possible to include a set of correct derivations in the training data. It is unlike most recent work on syntactic parsing in which gold-standard syntactic parses are available as the training data (Collins, 1997; Charniak, 2000; Clark & Curran, 2003). Therefore, the induction of a probabilistic model for derivations is a form of *unsupervised* learning. Figure 4 illustrates the overall semantic parsing framework of WASP.

Lexical learning will be the main focus of Section 3.2. Algorithms for inducing a probabilistic model will be described in Section 3.3.

3.2 Lexical Acquisition using Word Alignments

A lexicon is a mapping from words to their meanings. In Section 2.3, we showed that word alignments can be used for defining a mapping from words to their meanings. Using word alignment models to induce a lexicon is not a new idea (Manning & Schütze, 1999). Indeed, attempts have been made to directly apply machine translation systems to the problem of semantic parsing (Section 2.1.4). However, these systems make no use of the *formal grammar* of an MRL, thus allocating probability mass to MR translations that are not even syntactically well-formed. We argue that the MRL grammar can be a rich source of constraints for semantic analysis, and should not be left unused if available. In this section, we present an algorithm for inducing a lexicon from the training data, making sure that all MRs produced by the resulting parsing model are syntactically well-formed.

The basic idea of the lexical induction algorithm is to train a statistical word alignment model on the training data, and then find the most probable word alignments between the training sentences and their corresponding MRs. A lexicon is then formed by extracting transformation rules from these word alignments. Let us illustrate this algorithm using an example. Suppose that we are given as the training data the following English sentence and its meaning encoded in CLANG:

If our player 4 has the ball, our player 4 should shoot.
`((bowner our {4}) (do our {4} (shoot)))`

To train a word alignment model, we can feed the model with two input streams, one by lining up the English words, and the other by lining up the MR tokens. However, treating MR tokens as words is a bad idea for two reasons. First, not all MR tokens carry specific meanings. For example, in CLANG, parentheses (`(`, `)`) and braces (`{`, `}`) are delimiters that do not carry any specific meanings. Such tokens are not supposed to be aligned to any words, and inclusion of these tokens in the training data is likely to confuse the word alignment model. Second, MR tokens may exhibit polysemy. For instance, the CLANG predicate `pt` has three meanings based on the types of arguments it is given. It specifies the xy-coordinates when its arguments are two numbers (e.g. `(pt 0 0)`), the current position of the ball when its argument is the MR token `ball` (i.e. `(pt ball)`), or the current position of a player when a team and a uniform number are given as arguments (e.g. `(pt our 4)`). Given a token `pt`, the word alignment model would not be able to identify its exact meaning, unless its arguments are examined as well.

A simple, principled way to avoid these difficulties is to represent an MR by lining up the MRL productions used for generating it. More specifically, an MR parse tree is *linearized* by lining up the productions in the order of a top-down, left-most derivation. Figure 5 shows a sample word alignment between the above English sentence and its CLANG translation. Here the second production, CONDITION → `(bowner` TEAM `{`UNUM`})`, is the one that rewrites the CONDITION non-terminal in the first production, RULE → `(`CONDITION DIRECTIVE`)`, and so on. Treating MRL productions as words allows collocations to be treated as a single lexical unit (e.g. the tokens `(`, `shoot`, followed by `)`). Such collocations can be discontinuous (e.g. the parentheses in `(`CONDITION DIRECTIVE`)`). It also allows the meaning of a polysemous MR token to be disambiguated, because each possible meaning corresponds to a distinct MRL production, which can be easily identified provided that the MRL grammar is unambiguous. In addition, it allows productions that generate only non-terminals (e.g. $X \to YZ$) to be aligned to some English words. The only productions not included in a linearized parse tree are unary productions $X \to Y$, which often indicate a hypernym (is-a) relationship which is not usually realized in NL. For each unary production $X \to Y$, a rule $X \to \langle Y_{\boxed{1}}, Y_{\boxed{1}} \rangle$ is automatically added to the lexicon. Note that the structure of an MR parse tree is preserved through linearization, since for each linearized MR parse there is only one possible parse tree

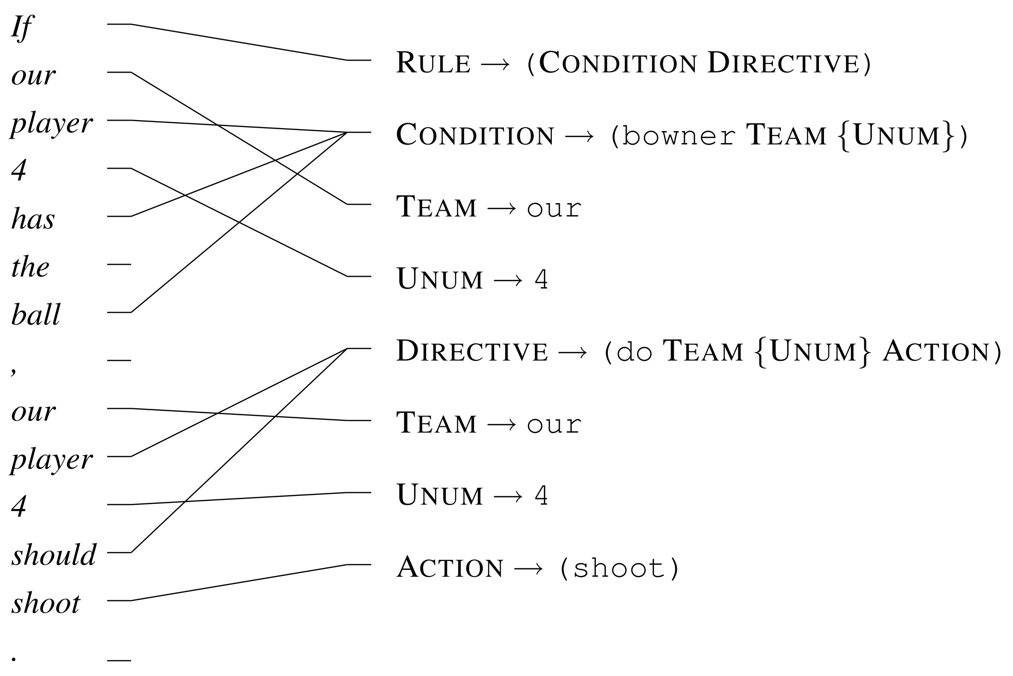


Figure 5: A sample alignment showing the correspondence between English words and CLANG productions

that corresponds to it, provided that the MRL grammar is unambiguous. The structural aspect of an MR parse tree will play an important role during the subsequent extraction of transformation rules. Although we choose to linearize an MR parse tree in the order of a top-down, left-most derivation, the exact order should not concern us, as long as the linearization preserves the structural aspect of a parse tree, because a good alignment model should be able to properly handle any differences in word order of a language pair.

Transformation rules are extracted from an alignment between NL words and MRL productions. The rule extraction algorithm assumes an n-to-1 alignment, where each word may contribute to at most one component of an MR. This is an assumption made by most existing semantic role labeling systems (Koomen et al., 2005; Toutanova et al., 2005), which has been found reasonable in most cases. Transformation rules are extracted in a bottom-up manner, which we illustrate using the alignment in Figure 5. The process starts with productions whose RHS is all terminals. There are five of them in the figure: two instances of TEAM → `our`, two instances of UNUM → `4`, and one instance of ACTION → `(shoot)`. Each of them is linked to exactly one word in the sentence. In this case, the extracted rules are obvious. The patterns consist of the word to which each production is linked, and the templates are the productions' RHS:

TEAM → ⟨ *our* , `our` ⟩
UNUM → ⟨ *4* , `4` ⟩
ACTION → ⟨ *shoot* , `(shoot)` ⟩

Next we consider the production CONDITION → `(bowner` TEAM `{`UNUM`})`. This production is linked to three words in the sentence: *player*, *has* and *ball*. Moreover, the predicate `bowner` has two arguments, a TEAM and a UNUM. These arguments correspond to the first instances of the productions TEAM → `our` and UNUM → `4`, which are linked to the words *our* and *4*, respectively. Here the alignment indicates not only how a predicate is realized in NL, but also how a *predicate-argument* relationship is realized:

```
Input: A training set $\{\langle \mathbf{e}_i, \mathbf{f}_i \rangle | i = 1, \ldots, N\}$, an unambiguous CFG $G'$ for the MRL, a word
       alignment model $T$, an integer $K > 0$.
Output: A lexicon $\mathcal{L}$ covering the training set.
begin
    $\mathcal{L} \leftarrow \emptyset$
    for $i \leftarrow 1$ to $N$ do
        $\mathbf{u} \leftarrow$ unique parse tree of $\mathbf{f}_i$ obtained by applying $G'$
        $\mathbf{r} \leftarrow$ list of productions of $G'$ obtained by linearizing $\mathbf{u}$ in the order of a top-down, left-most
        derivation
        $\mathbf{a}^\star_{1,\ldots,K} \leftarrow$ the $K$ most probable word alignments between $\mathbf{e}_i$ and $\mathbf{r}$ given by $T$
        for $k \leftarrow 1$ to $K$ do
            $C \leftarrow \emptyset$
            for $j \leftarrow |\mathbf{r}|$ downto $1$ do
                $\mathbf{w} \leftarrow$ list of words to which $r_j$ is linked according to $\mathbf{a}^\star_k$
                $\mathbf{r}' \leftarrow$ list of productions from $\mathbf{r}$ that rewrite the RHS non-terminals of $r_j$
                $\mathbf{p} \leftarrow$ pattern obtained by assembling $\mathbf{w}$ and $C(r')$ for all $r' \in \mathbf{r}'$, in the order they
                appear in $\mathbf{e}_i$, adding word gaps if necessary
                $\mathbf{t} \leftarrow \mathrm{rhs}(r_j)$
                Add indices to non-terminals in $\mathbf{p}$ and $\mathbf{t}$ to show their association
                Replace the substring of $\mathbf{e}_i$ covered by $\mathbf{p}$ with $\mathrm{lhs}(r_j)$, which is assigned to $C(r_j)$
                $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathrm{lhs}(r_j) \rightarrow \langle \mathbf{p}, \mathbf{t} \rangle\}$
end
```

Figure 6: The basic rule extraction algorithm based on word alignments

The NL substring referring to the TEAM argument (*our*) comes before the word *player*, and the substring referring to the UNUM argument (*4*) comes after it. Therefore, the pattern associated with the `bowner` production is TEAM *player* UNUM *has* (1) *ball*, where the word gap (1) is due to the unaligned word *the* that comes between *has* and *ball*. By adding indices to reflect the association between non-terminals in the pattern and the arguments of `bowner`, the following rule is extracted:

$$\text{CONDITION} \rightarrow \langle\ \text{TEAM}_{\boxed{1}}\ \textit{player}\ \text{UNUM}_{\boxed{2}}\ \textit{has}\ (1)\ \textit{ball}\ ,\ \texttt{(bowner}\ \text{TEAM}_{\boxed{1}}\ \texttt{\{}\text{UNUM}_{\boxed{2}}\texttt{\})}\ \rangle$$

Similarly, the following rules are extracted for the remaining productions:

$$\begin{aligned} \text{DIRECTIVE} &\rightarrow \langle\ \text{TEAM}_{\boxed{1}}\ \textit{player}\ \text{UNUM}_{\boxed{2}}\ \textit{should}\ \text{ACTION}_{\boxed{3}}\ ,\ \texttt{(do}\ \text{TEAM}_{\boxed{1}}\texttt{\{}\text{UNUM}_{\boxed{2}}\texttt{\}}\ \text{ACTION}_{\boxed{3}}\texttt{)}\ \rangle \\ \text{RULE} &\rightarrow \langle\ \textit{if}\ \text{CONDITION}_{\boxed{1}}\ (1)\ \text{DIRECTIVE}_{\boxed{2}}\ (1)\ ,\ \texttt{(}\text{CONDITION}_{\boxed{1}}\ \text{DIRECTIVE}_{\boxed{2}}\texttt{)}\ \rangle \end{aligned}$$

where the word gap (1) at the end of the last pattern is due to the unaligned period in the sentence. This word gap is added because all words in a sentence have to be consumed by a derivation.

Figure 6 shows the basic rule extraction algorithm of WASP. This algorithm requires a word alignment model T, which can be any off-the-shelf alignment model. In this work we choose IBM Model 5 (Brown et al., 1993) because efficient implementations of it are publicly available (Al-Onaizan et al., 1999; Och & Ney, 2003), and it tends to observe phrasal coherence in the restricted domains in which our semantic parser is evaluated (Sections 2.3.1 and 3.4.4). The training set, $\{\langle \mathbf{e}_i, \mathbf{f}_i \rangle\}$, is used for training the alignment model T, which is in turn used for obtaining the K-best word alignments for each example in the training set.

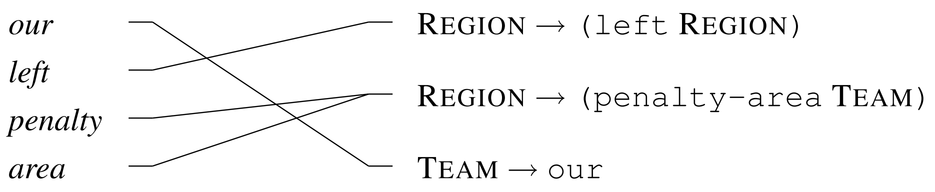


Figure 7: A sample English phrase and its CLANG representation, whose parse trees are not isomorphic

Transformation rules are extracted from each of these alignments. It is done in a bottom-up fashion, such that an MR predicate is processed only after its arguments have all been processed. This order is enforced by the backward traversal of the linearized MR parse, $\mathbf{r}$. The set C stores the mapping from productions to NL substrings, which is updated whenever a rule is extracted. Patterns are formed by assembling words and non-terminals that correspond to previously extracted patterns. The lexicon, $\mathcal{L}$, then consists of all rules extracted from all K-best word alignments based on all training examples.

3.2.1 Maintaining Parse Tree Isomorphism

There are two cases where the algorithm outlined in Figure 6 would not extract any rules for a production r:

1. None of the descendants of r in the MR parse tree are linked to any words.
2. The pattern for r covers a word w linked to a production r' that is *not* a descendant of r in the MR parse tree. Rule extraction is forbidden because it would destroy the link between w and r'.

The first case arises when a component of an MR is not realized. For example, the concept of *our team* is often assumed, because advice is given from the perspective of a team coach. When we say *the goalie should always stay in our goal area*, we mean our (`our`) goalie, not the other team's (`opp`) goalie. Hence the term `our` is often not realized. The second case arises when the NL and MR parse trees fail to be isomorphic. Consider the word alignment between *our left penalty area* and its MR, `(left (penalty-area our))` (Figure 7). Extracting the rule $\text{REGION} \rightarrow \langle \text{TEAM}_{\boxed{1}}\ (1)$ *penalty area*, $\texttt{(penalty-area}\ \text{TEAM}_{\boxed{1}}\texttt{)}\rangle$ would destroy the link between *left* and REGION → `(left` REGION`)`. A possible explanation for this is that, syntactically, *our* modifies *left penalty area* (consider the coordination phrase *our left penalty area and right goal area*, where *our* modifies both *left penalty area* and *right goal area*). But conceptually, *left* modifies the concept of *our penalty area* by referring to its left half. Note that the NL and MR parse trees must be isomorphic under the SCFG formalism (Section 2.2).

The NL and MR parse trees can be made isomorphic by merging nodes in the MR parse tree, combining several productions into one. In machine translation terminology, productions are merged to form a *bead* (Brown et al., 1991). For example, since no rules can be extracted for the production REGION → `(penalty-area` TEAM`)`, it is combined with its parent to form REGION → `(left (penalty-area` TEAM`))`, for which the pattern TEAM *left penalty area* is extracted. In general, the merging process continues until a rule is extracted from the merged node. Assuming the alignment is not empty, the process is guaranteed to end with a rule extracted.

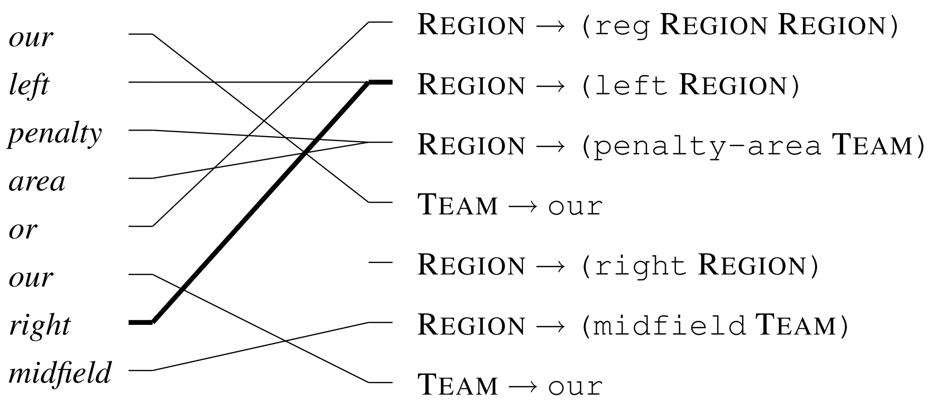


Figure 8: A moderate case of phrasal incoherence, where the link that causes it is shown as a thick line

3.2.2 Reducing Phrasal Incoherence

The effectiveness of the rule extraction algorithm described so far critically depends on whether the word alignment model observes phrasal coherence. This means words that are linked to an MR predicate and its arguments should stay close to each other. Moreover, these words should form a hierarchical structure that is roughly isomorphic to the MR parse tree. Any major disruption of this hierarchical structure would lead to long patterns and templates, a major cause of overfitting. For example, a single bad link in Figure 8 (shown as a thick line) would lead to the extraction of the rule REGION → ⟨TEAM$_{\boxed{1}}$ *left penalty area or* TEAM$_{\boxed{2}}$ *right midfield*, `(reg (left (penalty-area` TEAM$_{\boxed{1}}$`)) (right (midfield` TEAM$_{\boxed{2}}$`)))`⟩, which does not generalize well to other cases of region union (`reg`). This is not the worst case of phrasal incoherence. In the extreme, a single bad link could cause a pattern as long as a sentence to be extracted. Obviously, we need an alignment model that is aware of the formal syntax of the MRL, thereby maintaining phrasal coherence and allowing generally-applicable rules to be formed. However, this is a chicken and egg problem. To build a model that strictly observes phrasal coherence often requires transformation rules that model the reordering of tree nodes, as in most recent work on syntax-based alignment models (Section 2.3.2). Our goal is to bootstrap the learning process by using a simpler, word-based alignment model that produces an alignment that is generally coherent, and then refine it to recover any hierarchical structure of a sentence that has been obscured by bad links.

We refine an alignment, $\mathbf{a}$, by removing links that could lead to excessively long patterns and templates. Recall that rule extraction is forbidden for a production, r, if the pattern for r covers a word linked to a production that is outside the MR parse rooted at r. We call each such word a *violation* of the isomorphism constraint between NL and MR parse trees. For each production r in a linearized MR parse tree, we count the number of violations that would prevent a rule from being extracted for r. Then a total sum for all productions is obtained, denoted by $v(\mathbf{a})$. A simple procedure for removing bad links is to repeatedly remove a link $a \in \mathbf{a}$ that would maximize $v(\mathbf{a}) - v(\mathbf{a}\backslash\{a\}) > 0$, until $v(\mathbf{a})$ cannot be further reduced. A link stronger than a certain threshold (0.9) is never removed, so that merging of productions as in Figure 7 is still possible. The strength of a link $w \leftrightarrow r$ is defined as the translation probability, $\Pr(r|w)$, which is found to be highly correlated with the validity of a link. To replenish the removed links, links from a reverse alignment, $\tilde{\mathbf{a}}$ (obtained by treating the source language as target, and vice versa), are added to $\mathbf{a}$, as long as $\mathbf{a}$ remains n-to-1, and $v(\mathbf{a})$ is not increased.

The complete lexical induction algorithm is thus the following: Learn a word alignment model, T, and a reverse word alignment model, $\tilde{T}$, using a training set, $\{\langle \mathbf{e}_i, \mathbf{f}_i \rangle\}$. Obtain the K most probable alignments, $\mathbf{a}^\star_{1,\ldots,K}$, and the most probable reverse alignment, $\tilde{\mathbf{a}}^\star$, for each training example using T and $\tilde{T}$. Remove bad links from each $\mathbf{a}^\star_k$ and replenish the removed links by adding links from $\tilde{\mathbf{a}}^\star$. Then extract rules from $\mathbf{a}^\star_{1,\ldots,K}$ as described in Figure 6 and Section 3.2.1.

3.3 Parameter Estimation for the Semantic Parsing Model

3.3.1 A Probabilistic SCFG Model

Now that a lexicon is acquired, the next task is to learn a probabilistic model for the semantic parser. Since WASP is based on SCFG, an obvious, mathematically sound approach is probabilistic SCFG (PSCFG), which is a generalization of probabilistic CFG (PCFG) (Booth & Thompson, 1973; Jelinek & Lafferty, 1991). A PSCFG defines a probability distribution over the set of paired parse trees, one for NL and one for MRL. Each rule in a lexicon is assigned a probability of its use. Given a non-terminal, X, it is assumed that for all rules $r = X \to \langle \mathbf{p}, \mathbf{t} \rangle$, $\Pr_\theta(r|X)$ is a non-negative number and that:

$$\sum_r \Pr_\theta(r|X) = 1 \tag{10}$$

In addition, a special non-terminal, Γ, is introduced. Each Γ non-terminal may be rewritten to a single word. A word gap in a pattern is thus a shorthand for a finite number of Γ non-terminals, and the words generated from Γ are the ones being skipped over during pattern matching. For each word w, there is a special rule $\Gamma \to \langle w, \epsilon \rangle$ that writes to only the NL stream. Each special rule r_w is assigned a probability $\Pr_\theta(r_w|\Gamma)$. Like $\Pr_\theta(r|X)$, $\Pr_\theta(r_w|\Gamma)$ is a non-negative number and:

$$\sum_w \Pr_\theta(r_w|\Gamma) = 1 \tag{11}$$

Sentences and MRs are generated by repeatedly rewriting non-terminals using rules. The probability of a derivation, $\mathbf{d}$, is equal to the product of all rule probabilities involved, assuming that each rewriting decision is *independent*:

$$\Pr_\theta(\mathbf{d}) = \prod_{r \in \mathbf{d}} \Pr_\theta(r|\,\mathrm{lhs}(r))^{f_r(\mathbf{d})} \tag{12}$$

where $f_r(\mathbf{d})$ is the number of times a rule r is used in a derivation $\mathbf{d}$. The output MR, $\mathbf{f}^\star$, for an NL sentence $\mathbf{e}$ is thus the yield of the most probable derivation that yields $\mathbf{e}$ in the NL stream (Equation 9):

$$\mathbf{f}^\star = m\left(\arg\max_{\mathbf{d}} \Pr_\theta(\mathbf{d}, \mathbf{e})\right) = m\left(\arg\max_{\mathbf{d}} \prod_{r \in \mathbf{d}} \Pr_\theta(r|\,\mathrm{lhs}(r))^{f_r(\mathbf{d})}\right) \tag{13}$$

This is easily computed by the Viterbi algorithm (Viterbi, 1967; Jelinek, 1985). An Earley chart (Earley, 1970; Stolcke, 1995) can be used for keeping track of all possible (left-most) derivations that are consistent with the input up to a certain point. During parameter estimation, rule probabilities are found such that the joint likelihood of a training set $\{\langle \mathbf{e}_i, \mathbf{f}_i \rangle\}$ is maximized:

$$\theta^\star = \arg\max_\theta \prod_{\langle \mathbf{e}_i, \mathbf{f}_i \rangle} \sum_{\mathbf{d}} \Pr_\theta(\mathbf{d}, \mathbf{e}_i, \mathbf{f}_i) \tag{14}$$

If each training example were labeled with a correct derivation, then the maximum-likelihood estimate for rule probabilities would be the relative-frequency estimator. However, since correct derivations are not observed, an Expectation Maximization (EM) algorithm (Dempster et al., 1977) is used instead to find rule probabilities that locally maximize the likelihood of the training set, treating derivations as hidden variables. The Inside-Outside algorithm (Baker, 1979; Lari & Young, 1990) is an instance of the EM algorithm that uses the same chart as the Viterbi algorithm. The probabilities found may not be globally optimal, so the algorithm is sensitive to initial estimates. We assume as little as possible, using the uniform distribution as the initial estimate.

Rules with probability less than a certain threshold (e^{-100}) are discarded. Only rules that are used in the best parses for the training set are retained once the model converges. All other rules are discarded. The estimation process is repeated until no more rules are discarded. This heuristic is commonly known as *Viterbi approximation*, and is used to avoid erroneous parses, assuming that rules that are used in the best parses are the most accurate ones (Zettlemoyer & Collins, 2005). To deal with unseen words w for which special word gap rules $\Gamma \to \langle w, \epsilon \rangle$ are not learned, the uniform distribution $\Pr_u(r_w|\Gamma) = 1/|\mathcal{T}_n|$ is used as a back-off model, where $|\mathcal{T}_n|$ is the size of the training vocabulary.

Unsupervised induction of generative models based on PCFG have been used for syntactic parsing (Pereira & Shabes, 1992; Stolcke & Omohundro, 1994; Klein & Manning, 2004), language modeling (Baker, 1979; Chen, 1995), and for providing evidence against the poverty of the stimulus (Clark, 2001). Use of PSCFG for machine translation has been suggested by Melamed (2004). However, for semantic parsing, PSCFG has a few potential problems:

1. PSCFG has a strong bias toward small parse trees (Manning & Schütze, 1999). This is because rewritings can only reduce the overall parse probability (Equation 10). A bias toward small parse trees means rules having long patterns and templates are favored over shorter ones, which could hurt the model's ability to generalize.

2. To determine the best parse, only a *conditional* parse probability, $\Pr(\mathbf{d}|\mathbf{e})$, is required, since the probability of observation, $\Pr(\mathbf{e})$, is fixed. It seems reasonable to directly estimate a conditional probability distribution, rather than spending modeling effort on observations.

3.3.2 A Maximum-Entropy Model

This section presents a maximum-entropy model for WASP. Unlike PSCFG, a maximum-entropy model directly defines a conditional probability distribution over paired parse trees, given the observed data. It is also known as log-linear models (Knoke & Burke, 1980) and random fields (Geman & Geman, 1984). Fully-supervised maximum-entropy models have been applied successfully to part-of-speech tagging (Ratnaparkhi, 1996; Lafferty et al., 2001), syntactic parsing (Ratnaparkhi, 1999; Charniak, 2000; Clark & Curran, 2003), text classification (Taskar et al., 2002), and named entity recognition (Chieu & Ng, 2003). In contrast, little work has been done in NLP that uses maximum-entropy models with incomplete data (Riezler et al., 2000; Zettlemoyer & Collins, 2005). We argue that such models can be useful in semantic parsing.

A maximum-entropy model is an exponential model:

$$\Pr_\alpha(\mathbf{d}|\mathbf{e}) = \frac{1}{Z_\alpha(\mathbf{e})} \prod_i \alpha_i^{f_i(\mathbf{d})} \tag{15}$$

where the conditional probability, $\Pr_\alpha(\mathbf{d}|\mathbf{e})$, is proportional to the product of weights α_i assigned to each feature f_i. A *feature* represents a certain characteristic of a derivation. In this case, the features are the number of times each transformation rule is used in a derivation. (Note that in PSCFG, the probability of a derivation is also determined by the number of times each transformation rule is used.) The function $Z_\alpha(\mathbf{e})$, called a partition function, is a normalizing factor such that the conditional probabilities sum to one over all derivations that yield $\mathbf{e}$. Clearly, PSCFG is also an exponential model (cf. Equation 12). The difference is that, in a maximum-entropy model, features may interact with each other, so the independence assumptions (Equations 10 and 11) need not hold. A consequence is that feature weights, α_i, can be any positive numbers, whereas in PSCFG, $\Pr_\theta(r|\,\mathrm{lhs}(r))$ must not exceed one. Since rewritings may actually increase the overall probability of a derivation, there is no bias toward small parse trees. Moreover, smoothing can be done in a more principled manner (Charniak, 2000). Recall that in PSCFG, a back-off model is used to deal with unseen words generated from word gaps. In a maximum-entropy model, generation of unseen words can be modeled using an extra feature, $f_*(\mathbf{d})$, whose value is the number of *all* words being skipped. Additional features that correspond to domain-specific word classes can be used for more fine-grained smoothing. The fact that these features may interact with each other is not a concern.

Decoding of a maximum-entropy model can be done using the same algorithm as PSCFG (cf. Equation 13):

$$\mathbf{f}^\star = m\left(\arg\max_{\mathbf{d}} \Pr_\alpha(\mathbf{d}|\mathbf{e})\right) = m\left(\arg\max_{\mathbf{d}} \prod_i \alpha_i^{f_i(\mathbf{d})}\right) \tag{16}$$

The maximum conditional likelihood criterion is used for estimating a maximum-entropy model (Berger et al., 1996; Johnson et al., 1999). This means that the conditional likelihood of $\mathbf{f}_i$ given $\mathbf{e}_i$ is maximized, instead of the joint likelihood of $\mathbf{e}_i$ and $\mathbf{f}_i$ as in PSCFG (cf. Equation 14). This criterion is chosen because it is much easier to work with (calculation of the join likelihood would require summation over all possible parses, $D(G)$), and it allows for a form of *discriminative* learning that focuses on separating good parses from bad ones. Berger et al. (1996) also argues for this criterion from a maximum-entropy perspective. The conditional log-likelihood of a training set is as follows:

$$\begin{aligned}
& \sum_{\langle \mathbf{e}_j,\mathbf{f}_j\rangle} \log \Pr_\alpha(\mathbf{f}_j|\mathbf{e}_j) \\
= & \sum_{\langle \mathbf{e}_j,\mathbf{f}_j\rangle} \log \sum_{\mathbf{d}\in D(G|\mathbf{e}_j,\mathbf{f}_j)} \Pr_\alpha(\mathbf{d}|\mathbf{e}_j) \\
= & \sum_{\langle \mathbf{e}_j,\mathbf{f}_j\rangle} \left(\log\left(\sum_{\mathbf{d}\in D(G|\mathbf{e}_j,\mathbf{f}_j)} \prod_i \alpha_i^{f_i(\mathbf{d})}\right) - \log Z_\alpha(\mathbf{e}_j)\right) \\
= & \sum_{\langle \mathbf{e}_j,\mathbf{f}_j\rangle} \left(\log\left(\sum_{\mathbf{d}\in D(G|\mathbf{e}_j,\mathbf{f}_j)} \exp\sum_i \lambda_i f_i(\mathbf{d})\right) - \log\left(\sum_{\mathbf{d}\in D(G|\mathbf{e}_j)} \exp\sum_i \lambda_i f_i(\mathbf{d})\right)\right)
\end{aligned} \tag{17}$$

where λ_i is the logarithm of α_i, $D(G|\mathbf{e}_j)$ is the set of valid derivations that yield $\mathbf{e}_j$, and $D(G|\mathbf{e}_j, \mathbf{f}_j)$ is the set of valid derivations that yield both $\mathbf{e}_j$ and $\mathbf{f}_j$ (hence $D(G|\mathbf{e}_j, \mathbf{f}_j) \subseteq D(G|\mathbf{e}_j)$). Differentiating (17) with respect to λ_i gives:

$$\sum_{\langle \mathbf{e}_j,\mathbf{f}_j\rangle} \left(\sum_{\mathbf{d}\in D(G|\mathbf{e}_j,\mathbf{f}_j)} \Pr_\alpha(\mathbf{d}|\mathbf{e}_j,\mathbf{f}_j) f_i(\mathbf{d}) - \sum_{\mathbf{d}\in D(G|\mathbf{e}_j)} \Pr_\alpha(\mathbf{d}|\mathbf{e}_j) f_i(\mathbf{d})\right) \tag{18}$$

which is the difference between the expectations of $f_i(\mathbf{d})$ with respect to the distributions $\Pr_\alpha(\mathbf{d}|\mathbf{e}_j, \mathbf{f}_j)$ and $\Pr_\alpha(\mathbf{d}|\mathbf{e}_j)$. Setting (18) to zero yields the condition for an extremum of the conditional log-likelihood with respect to a single parameter, λ_i. However, since (18) depends on all α_i (and hence all λ_i), the system of equations cannot be solved coordinate-wise.

To find a set of parameters $\lambda^\star$ that (locally) maximize the conditional log-likelihood, we use a version of *improved iterative scaling* (Della Pietra et al., 1997) which has been used for estimating probabilistic unification-based grammars (Riezler et al., 2000). The main idea is to find a new set of parameters $\lambda + \delta$ given the initial parameters, λ, such that the conditional log-likelihood does not decrease. If a procedure that maps λ to $\lambda + \delta$ can be found, then the procedure can be applied until a fixed point $\lambda^\star$ is reached.

Consider the change in conditional log-likelihood from λ to $\lambda + \delta$:

$$\begin{aligned}
\Delta L_\lambda(\delta) &= \sum_{\langle \mathbf{e}_j,\mathbf{f}_j\rangle} \left(\log \Pr_{\lambda+\delta}(\mathbf{f}_j|\mathbf{e}_j) - \log \Pr_\lambda(\mathbf{f}_j|\mathbf{e}_j)\right) \\
&= \sum_{\langle \mathbf{e}_j,\mathbf{f}_j\rangle} \left(\log \frac{Z_{\lambda+\delta}(\mathbf{e}_j,\mathbf{f}_j)}{Z_\lambda(\mathbf{e}_j,\mathbf{f}_j)} - \log \frac{Z_{\lambda+\delta}(\mathbf{e}_j)}{Z_\lambda(\mathbf{e}_j)}\right) \\
&= \sum_{\langle \mathbf{e}_j,\mathbf{f}_j\rangle} \left(\left(\log \sum_{\mathbf{d}\in D(G|\mathbf{e}_j,\mathbf{f}_j)} \frac{P_\lambda(\mathbf{d})}{Z_\lambda(\mathbf{e}_j,\mathbf{f}_j)} \frac{P_{\lambda+\delta}(\mathbf{d})}{P_\lambda(\mathbf{d})}\right) - \log \frac{Z_{\lambda+\delta}(\mathbf{e}_j)}{Z_\lambda(\mathbf{e}_j)}\right)
\end{aligned}$$

where $P_\lambda(\mathbf{d})$ is the *unnormalized* probability, $\exp \sum_i \lambda_i f_i(\mathbf{d})$, of a derivation $\mathbf{d}$. Since $P_\lambda(\mathbf{d})/Z_\lambda(\mathbf{e}_j,\mathbf{f}_j)$ is a probability distribution over all derivations $\mathbf{d}$ that yield $\mathbf{e}_j$ and $\mathbf{f}_j$, Jensen's inequality can be applied:

$$\begin{aligned}
&\Delta L_\lambda(\delta) \\
\geq &\sum_{\langle \mathbf{e}_j,\mathbf{f}_j\rangle} \left(\left(\sum_{\mathbf{d}\in D(G|\mathbf{e}_j,\mathbf{f}_j)} \frac{P_\lambda(\mathbf{d})}{Z_\lambda(\mathbf{e}_j,\mathbf{f}_j)} \log \frac{P_{\lambda+\delta}(\mathbf{d})}{P_\lambda(\mathbf{d})}\right) - \log \frac{Z_{\lambda+\delta}(\mathbf{e}_j)}{Z_\lambda(\mathbf{e}_j)}\right) \\
\geq &\sum_{\langle \mathbf{e}_j,\mathbf{f}_j\rangle} \left(\left(\sum_{\mathbf{d}\in D(G|\mathbf{e}_j,\mathbf{f}_j)} \Pr_\lambda(\mathbf{d}|\mathbf{e}_j,\mathbf{f}_j) \sum_i \delta_i f_i(\mathbf{d})\right) + \left(1 - \frac{Z_{\lambda+\delta}(\mathbf{e}_j)}{Z_\lambda(\mathbf{e}_j)}\right)\right) \\
= &\sum_{\langle \mathbf{e}_j,\mathbf{f}_j\rangle} \left(\left(\sum_{\mathbf{d}\in D(G|\mathbf{e}_j,\mathbf{f}_j)} \Pr_\lambda(\mathbf{d}|\mathbf{e}_j,\mathbf{f}_j) \sum_i \delta_i f_i(\mathbf{d})\right) + 1 - \left(\sum_{\mathbf{d}\in D(G|\mathbf{e}_j)} \Pr_\lambda(\mathbf{d}|\mathbf{e}_j) \exp \sum_i \delta_i f_i(\mathbf{d})\right)\right) \\
\geq &\sum_{\langle \mathbf{e}_j,\mathbf{f}_j\rangle} \left(\left(\sum_{\mathbf{d}\in D(G|\mathbf{e}_j,\mathbf{f}_j)} \Pr_\lambda(\mathbf{d}|\mathbf{e}_j,\mathbf{f}_j) \sum_i \delta_i f_i(\mathbf{d})\right) + 1 \right. \\
&\qquad \left. - \left(\sum_{\mathbf{d}\in D(G|\mathbf{e}_j)} \Pr_\lambda(\mathbf{d}|\mathbf{e}_j) \sum_i \left(\frac{f_i(\mathbf{d})}{f^\sharp(\mathbf{d})}\right) \exp \delta_i f^\sharp(\mathbf{d})\right)\right) \qquad (19)
\end{aligned}$$

The second inequality is due to $-\log \alpha \geq 1 - \alpha$, which is true for all $\alpha > 0$. Then assuming $f^\sharp(\mathbf{d}) = \sum_i f_i(\mathbf{d})$, Jensen's inequality is applied, yielding the lower bound in (19). We call this lower bound $A_\lambda(\delta)$. Differentiating A_λ with respect to δ_i gives:

$$\sum_{\langle \mathbf{e}_j,\mathbf{f}_j\rangle} \left(\left(\sum_{\mathbf{d}\in D(G|\mathbf{e}_j,\mathbf{f}_j)} \Pr_\lambda(\mathbf{d}|\mathbf{e}_j,\mathbf{f}_j) f_i(\mathbf{d})\right) - \left(\sum_{\mathbf{d}\in D(G|\mathbf{e}_j)} \Pr_\lambda(\mathbf{d}|\mathbf{e}_j) f_i(\mathbf{d}) \exp \delta_i f^\sharp(\mathbf{d})\right)\right) \qquad (20)$$

in which the only free parameter is δ_i. Thus the parameters $\delta^\star$ that maximize A_λ (and hence increase the conditional log-likelihood of the training set) can be found by solving $\partial A_\lambda / \partial \delta_i = 0$ for each δ_i using the Newton-Raphson method.

Calculation of the first and second partial derivatives of A_λ requires statistics that depend on all parses in $D(G|\mathbf{e}_j, \mathbf{f}_j)$ and $D(G|\mathbf{e}_j)$ (see Riezler (1998) for details). Since both sets can be extremely large, it is not feasible to enumerate all of them. Fortunately, using the same chart used for estimating a PSCFG model, it is possible to obtain the required statistics using dynamic-programming techniques similar to the Inside-Outside algorithm (Miyao & Tsujii, 2002; Geman & Johnson, 2002).

A Gaussian prior is used for regularizing a maximum-entropy model, resulting in an additional term $-\delta_i$ in $\partial A_\lambda / \partial \delta_i$. Unlike the fully-supervised case, the conditional log-likelihood is not concave with respect to λ, so the estimation algorithm is sensitive to initial parameters. To assume as little as possible, all λ_i are set to zero initially. As described in the previous section, rules that are not used in the best parses for the training set are discarded in order to improve precision. However, since a maximum-entropy model takes longer to converge, rules are discarded before a fixed point is reached (every 10 iterations in our experiments).

In addition to improved iterative scaling, other optimization methods for general functions can be used for calculating $\lambda^\star$. These methods include gradient ascent, conjugate gradient (Fletcher & Reeves, 1964), and quasi-Newton methods (Byrd et al., 1994). Studies have indicated that quasi-Newton methods can outperform traditional iterative scaling approaches in terms of convergence rate and robustness (Malouf, 2002). We plan to explore these optimization methods in future.

In summary, the WASP semantic parsing framework consists of two main components. First is a lexical acquisition component, which is based on word alignments between NL sentences and linearized MR parses, given by an off-the-shelf word alignment model trained on a set of training examples. The extracted transformation rules form an SCFG, for which a probabilistic model is learned to resolve parse ambiguity. The second component of WASP is for estimating the parameters of a probabilistic model. Two parametric models are proposed, one based on PSCFG, the other one based on maximum entropy. The probabilistic model is trained on the same set of training examples in an unsupervised manner.

3.4 Experiments

We evaluated WASP in two domains. The first domain is ROBOCUP (Section 2.1.1). To build a corpus for this domain, 300 pieces of coaching advice coded in CLANG were randomly selected from the log files of the 2003 ROBOCUP Coach Competition. Each formal instruction was then manually translated into English by one of four annotators. The second domain is GEOQUERY. To collect data for this domain, 250 English questions were gathered from an undergraduate language class who had no prior knowledge of the database structure. These questions were then manually translated into a functional query language, resulting in a 250-example data set. An additional 630 English questions were subsequently gathered from an undergraduate AI class, and from users of a web interface to a CHILL prototype (Zelle & Mooney, 1996) trained on the 250 data set. These questions together with their functional language translations and the original data set, formed a larger 880-example data set. Queries in the 250 data set were also translated into Spanish and Turkish, each by a native speaker of the language, and into Japanese by an English speaker who learned Japanese as a second language. Figure 9 shows the corpus statistics. WASP has not been evaluated in the ATIS domain, but there is evidence that ATIS is a less challeging domain than GEOQUERY (Section 2.1.4).

Each data set was divided into 10 equal-sized subsets. Standard 10-fold cross validation was used for estimating how well a learned parser would perform on unseen data. For each of the 10 trials, one of the 10 subsets were used as a test set, and the remaining 9 subsets were put together to form a training set. A

Domain	ROBOCUP	GEOQUERY				
MRL	CLANG	functional GEOQUERY language				
No. of non-terminals	37	13				
No. of productions	133	137				
NL	English	English	Spanish	Japanese	Turkish	English
No. of examples	300	250				880
Avg. MR length (tokens)	13.42	6.20				6.47
Avg. NL sentence length	22.52	6.76	7.29	9.04	5.65	7.48
No. of unique NL tokens	337	159	157	155	216	270

Figure 9: Corpora used for evaluating semantic parsers

semantic parser was learned using the training set. The learned parser was then used for transforming NL sentences in the test set into MRs. Transformation failed when there were NL and MRL constructs in the test set that the learned parser did not cover. We counted the number of sentences that were completely transformed, and the number of translations that were correct. For CLANG, a translation was correct if it exactly matched the correct MR, up to reordering of the arguments of commutative predicates like `and`. For GEOQUERY, a translation was correct if it retrieved the same answer as the correct query. These rather stringent criteria were adopted because a slightly modified MR could mean something very different, so any partial correctness metrics would be misleading. Using these counts, we measured the performance of the parser in terms of *precision* and *recall*:

$$\text{Precision} = \frac{\text{No. of correct translations}}{\text{No. of completely transformed sentences}} \tag{21}$$

$$\text{Recall} = \frac{\text{No. of correct translations}}{\text{No. of sentences in test set}} \tag{22}$$

These statistics were averaged across all 10 trials.

For lexical learning, we used Och and Ney's (2003) implementation of IBM Model 5, GIZA++, for training word alignment models. IBM Models 1–4 were used for initializing the model parameters during training.

For each domain, there was a minimal set of *initial rules* representing knowledge needed for the most basic transformations. These rules were always included in a lexicon, regardless of training data. For CLANG, the initial rules were the following:

UNUM $\rightarrow \langle i, i \rangle$, for all integers $i = 1, \ldots, 11$
NUM $\rightarrow \langle x, x \rangle$, for all $x \in \mathbb{R}$
CLANGSTR $\rightarrow \langle w, \texttt{"}w\texttt{"} \rangle$, for all words w that can be a CLANG identifier (e.g. words that are neither numbers nor reserved words)

The purpose of these initial rules was to provide a default transformation for those numbers and CLANG identifiers that were not encountered during training. Note that the same pattern might refer to different entities. For example, for all integers $i = 1, \ldots, 11$, i could be either a UNUM (uniform number) or a NUM (real number). It was up to the semantic parser to disambiguate between these two cases based on surrounding context. Also the list of initial rules were not meant to be exhaustive. Additional rules could be learned for those alternative expressions referring to the same entities, e.g. the term *goalkeeper* for the

Parser	WASP	COCKTAIL	SCISSOR	Zettlemoyer et al. (2005)
Precision/recall (%) for ROBOCUP	88.85/61.93	–	89.5/73.7	–
Precision/recall (%) for GEOQUERY	86.14/75.00	89.92/79.40	91.5/72.3	96.25/79.29

Figure 10: Precision and recall for various semantic parsers at the end of learning curves

uniform number `1`. These initial rules could even be wrong or overly-general. It was up to the parameter estimator to determine the contribution of these rules, which could be negative. For GEOQUERY, similar initial rules were devised:

NUM → $\langle x, x \rangle$, for all $x \in \mathbb{R}$
CITY → $\langle \tau(s),$ `cityid('`s`',_)` $\rangle$, for all city names s (e.g. *new york*, *los angeles*); $\tau(s)$ is an NL translation of s (similar rules for other types of names, e.g. rivers)

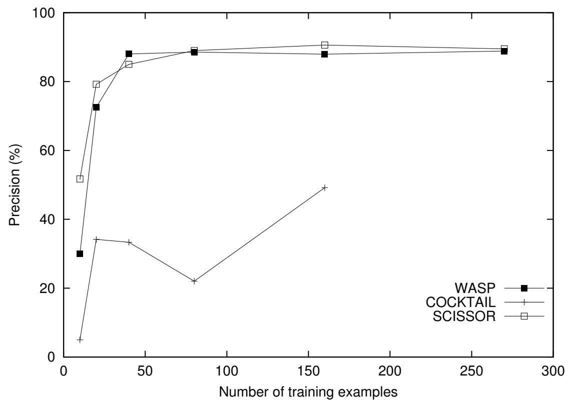
Since the geographical database was in English, $\tau(s) = s$ for English. For other languages, $\tau(s)$ could be different. For example, *New York* was transcribed as *Nyuu Yooku* in Japanese. The systematic change in vowels was due to various phonotactic constraints of the Japanese language. A mapping from *Nyuu Yooku* to `cityid('new york',_)` was thus made an initial rule, providing a semantic parser with domain knowledge that could not be easily learned without analyzing the phonological features of a name. Such initial rules were constructed from a bilingual dictionary. Note that a name could be ambiguous. For example, *New York* could be either a state or a city. Again, it was up to the semantic parser to disambiguate between these two cases based on surrounding context.

3.4.1 Comparison of Semantic Parsing Algorithms

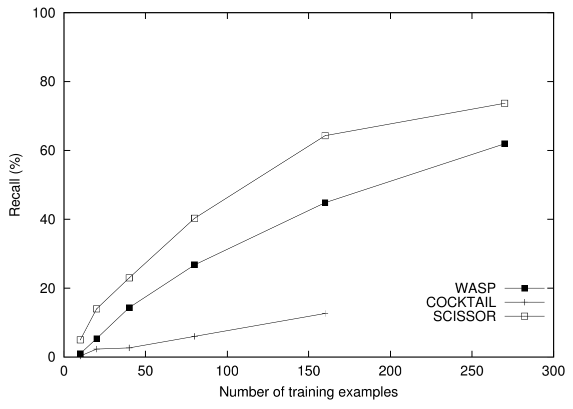
Figures 10 and 11 show the performance of WASP in the ROBOCUP and GEOQUERY domains, compared to three other algorithms: COCKTAIL is an unsupervised shift-reduce parser based on inductive logic programming (Tang & Mooney, 2001); SCISSOR is a fully-supervised combined syntactic-semantic parser based on PCFG (Ge & Mooney, 2005); Zettlemoyer and Collins (2005) is a partially-supervised combined syntactic-semantic parser based on combinatory categorial grammars (CCG). SCISSOR is fully-supervised because it requires full syntactic parses with semantic labels as the training data. Zettlemoyer and Collins's parser is partially-supervised in the sense that while neither syntactic nor semantic parses are included in training data, hand-built rules are required for building a CCG lexicon. These rules are specific to a particular NL and MRL pair. Parameters of the resulting CCG model are then estimated in an unsupervised manner. Note that while WASP, COCKTAIL and SCISSOR were evaluated by performing 10-fold cross validation using the same splits between training and test data, Zettlemoyer and Collins (2005) used a different experimental set-up, in which 600 GEOQUERY examples were explicitly set aside for training, 280 were used for testing, and the experiment was repeated twice to obtain average statistics. No results on ROBOCUP were reported in Zettlemoyer and Collins (2005).

Experimental results clearly show the advantage of extra supervision. For GEOQUERY, Zettlemoyer and Collins's parser has the highest precision and recall, although WASP remains competitive. For ROBOCUP, SCISSOR maintains a 10–20% lead over WASP throughout the learning curve in terms of recall. Note that the ROBOCUP domain is characterized by a smaller training set and longer utterances which lead to higher ambiguity (Figure 9). Supervision is particularly useful in these situations.

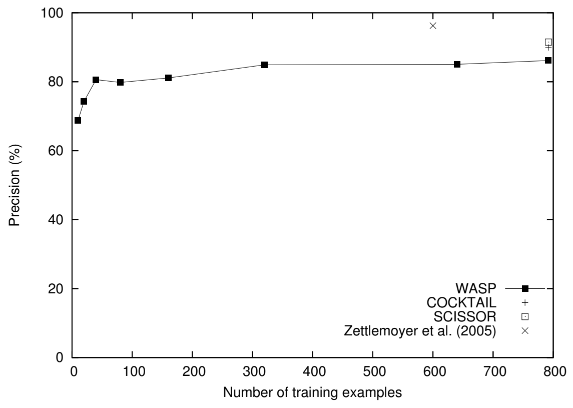
While COCKTAIL is highly competitive in the GEOQUERY domain, it does much worse in ROBOCUP. For ROBOCUP, it cannot handle training sets larger than 160 examples due to lack of memory, and its pre-



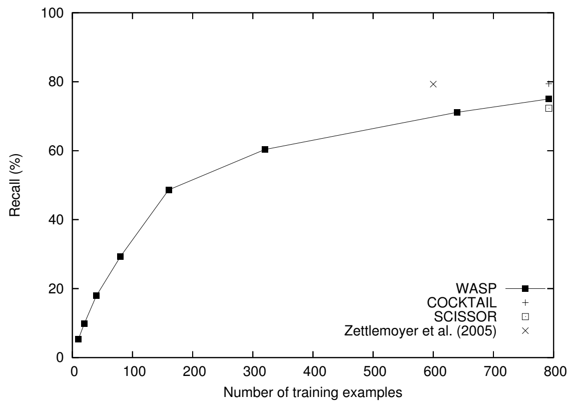
(a) Precision curves for ROBOCUP



(b) Recall curves for ROBOCUP



(c) Precision curves for GEOQUERY



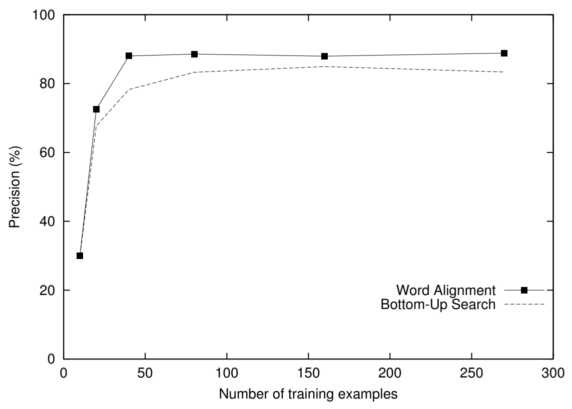
(d) Recall curves for GEOQUERY

Figure 11: Precision and recall curves comparing various semantic parsers

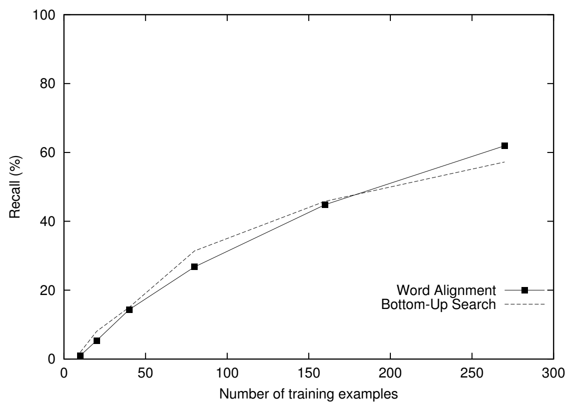
cision and recall are much lower. Apparently, the difficulty lies in the length of utterances being processed. COCKTAIL's deterministic shift-reduce framework is able to process a sentence only from beginning to end. So if it fails to parse the beginning of a sentence, then it will fail to parse the rest of the sentence. In contrast, WASP takes a holistic view of a sentence, allowing a decision made later to influence those made earlier, which proves necessary for more complex utterances.

3.4.2 Comparison of Lexical Learning Methods

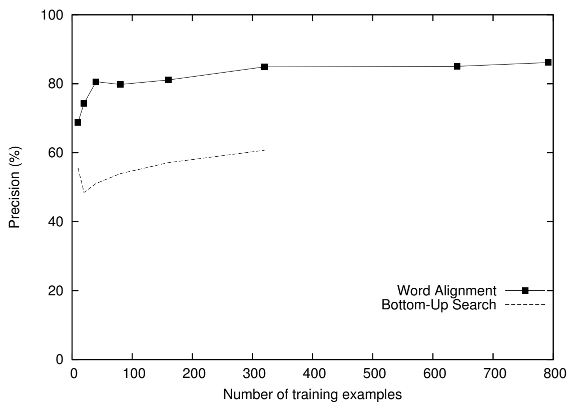
Beginning with this section, we evaluate the individual components of WASP. The aim of this section is to verify the usefulness of alignment models in lexical learning. To this end, we compare our alignment-based rule extraction algorithm against the bottom-up search algorithm presented in Kate et al. (2005). The idea of bottom-up search is to start with maximally-specific rules for each production r in the MRL grammar. A rule is maximally-specific when its pattern is a complete NL sentence. A maximally-specific rule is constructed for each positive example of r (i.e. whose MR contains an instance of r in its parse tree). These rules



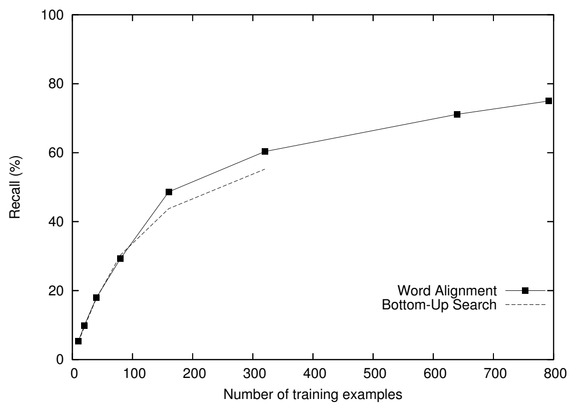
(a) Precision curves for ROBOCUP



(b) Recall curves for ROBOCUP



(c) Precision curves for GEOQUERY

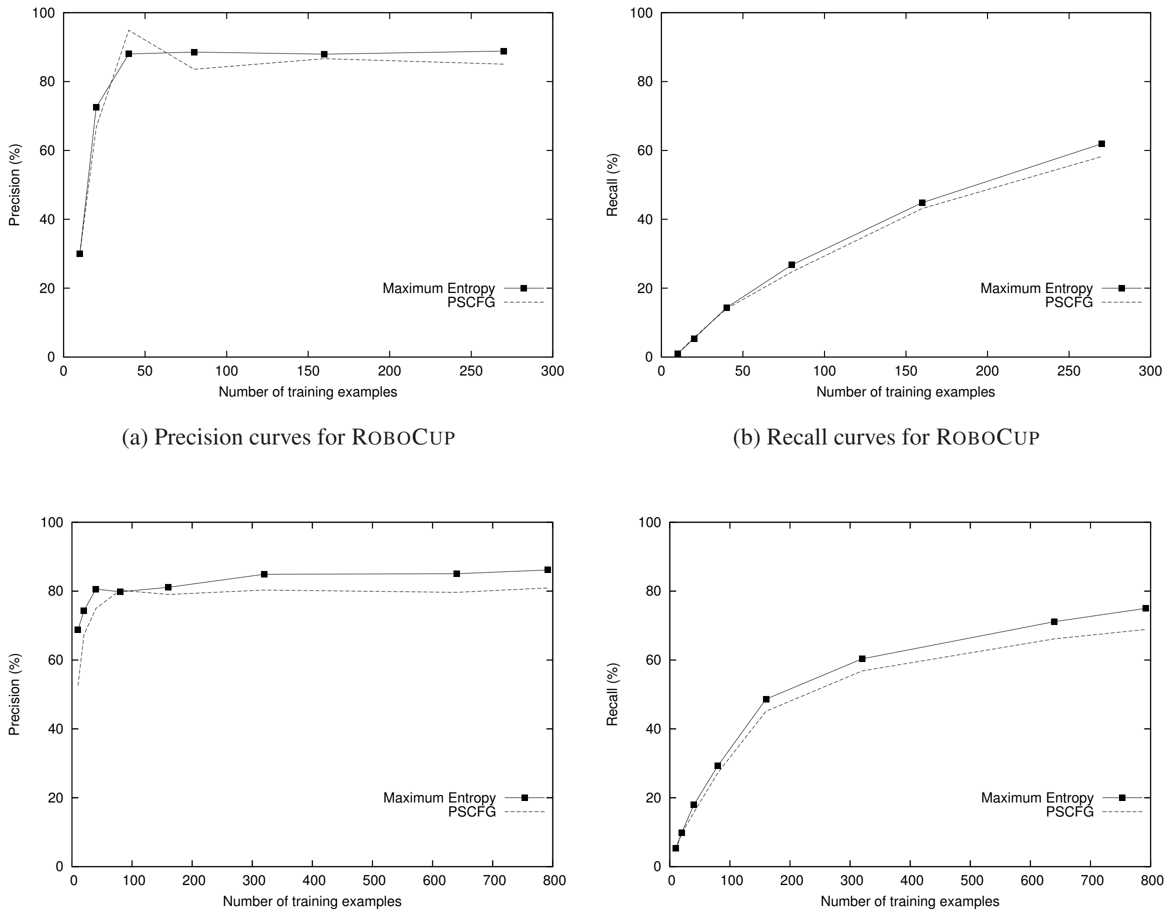


(d) Recall curves for GEOQUERY

Figure 12: Precision and recall curves comparing various lexical learning methods; maximum-entropy models are used for parameter estimation

are repeatedly generalized by taking their longest common subsequences until they start to cover too many negative examples. A lexicon consists of all generalized rules for all productions in the MRL grammar.

A major weakness of bottom-up search is that it is a local search that ignores all productions other than the one under consideration. It tends to produce rules that cover portions of a sentence that have other meanings. It contrasts with an alignment model that performs a global search to find alignments that are overall the best. To make up for this lack of global information, a lexicon is made to include as many rules as possible, in the hope that a subset of them will cooperate with each other to produce meaningful parses. Such a lexicon necessarily contains many irrelevant rules to be pruned away at a later stage. This proves to be a highly demanding task. As shown in Figure 12, a lexicon produced by bottom-up search is significantly less precise than one that is extracted from the best alignments, given the same level of recall. This indicates that not all irrelevant rules were pruned away during parameter estimation. In fact the lexicon grew so large in our GEOQUERY experiments that it took more than one day for the training algorithm to finish when the



(a) Precision curves for RoboCup

(b) Recall curves for RoboCup

(c) Precision curves for Geoquery

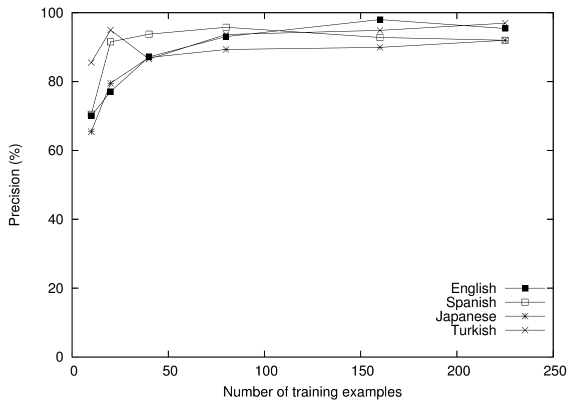
(d) Recall curves for Geoquery

Figure 13: Precision and recall curves comparing various probabilistic models; word alignment models are used for lexical learning

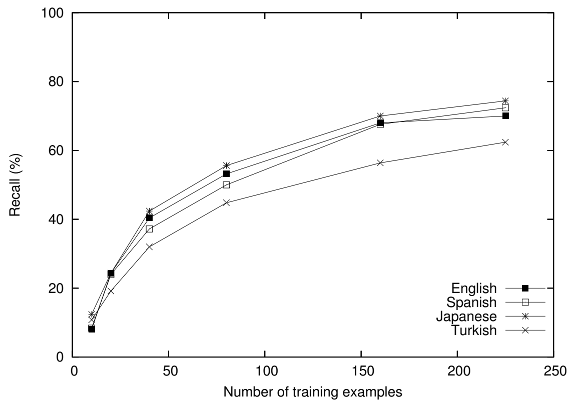
training size was more than 300. In contrast, training took no more than 90 minutes when an alignment model was used for lexical learning with a training size of 792. The usefulness of an alignment model thus lies in its ability to perform efficient global search, making sure the extracted rules will cooperate to produce complete parses.

3.4.3 Comparison of Probabilistic Models

The next component to evaluate is the probabilistic model for resolving parse ambiguity. Section 3.3 presents two probabilistic models based on PSCFG and maximum entropy, and Section 3.3.1 predicts that PSCFG will favor rules with longer patterns and templates, leading to lower coverage. Figure 13 supports this prediction, showing a slight decrease in recall when a PSCFG model is used. The decrease is statistically significant in both domains ($p = 0.019$ for RoboCup, $p < 0.001$ for Geoquery). Precision decreases by the same amount as inferior parses rise to the top for those examples that are not covered. Rule templates



(a) Precision curves for GEOQUERY



(b) Recall curves for GEOQUERY

Figure 14: Precision and recall curves comparing various natural languages

are on average 15–16% longer for a PSCFG model, and patterns are 29–30% longer.

The better performance of a maximum-entropy model is also attributable to its use of discriminative training, which directly optimizes the conditional probability of correct translations given an input sentence. This result is consistent with recent work on discriminative training of syntactic parsing models, which suggests that discriminative training alone can improve performance (Collins & Roark, 2004).

3.4.4 Comparison of Natural Languages

We conclude Section 3.4 by evaluating WASP in a variety of natural languages. The languages being considered are English, Spanish, Japanese and Turkish. These languages differ in terms of word order: Subject-Verb-Object (SVO) for English and Spanish, and Subject-Object-Verb (SOV) for Japanese and Turkish. Both English and Spanish are inflected languages, while Japanese and Turkish are agglutinative languages, where words are formed by joining morphemes together with a high morpheme-to-word ratio. Each combination of morphemes creates a different word. As a result, the Turkish GEOQUERY corpus contains 36% more unique words than the other GEOQUERY corpora of the same size (Figure 9). On the other hand, morphemes are separated into tokens in the Japanese corpus. So the Japanese sentences are the longest on average in terms of tokens (24% more than Spanish, the second longest).

Figure 14 shows the performance of WASP in various languages in the GEOQUERY domain. Since the larger vocabulary leads to less general rules, recall is the lowest in Turkish, and its precision is among the highest. The Japanese corpus has the lowest precision, presumably due to confusion brought about by the separated functional morphemes. There are no consistent differences between English and Spanish. More importantly, the performance is similar for different word order. It should come as no surprise, since WASP assumes nothing about word order. In particular, the alignment model (IBM Model 5) seems to handle differences in word order very well. This result suggests that word order (and hence the order in which MR parse trees are linearized) is not a major concern, at least in restricted domains such as GEOQUERY.

4 Proposed Research

In this section, we discuss ongoing and future research work that extends the WASP algorithm in various ways. Section 4.1 describes how prior knowledge about the NL syntax can be exploited in WASP. Section 4.2 turns to prior knowledge about the MRL and the application domain. Section 4.3 motivates the use of syntax-aware word alignment models for lexical learning. Section 4.4 explores the possibility of extending WASP to deal with various issues such as anaphora resolution and referential ambiguity.

4.1 Utilizing Syntactic Annotations

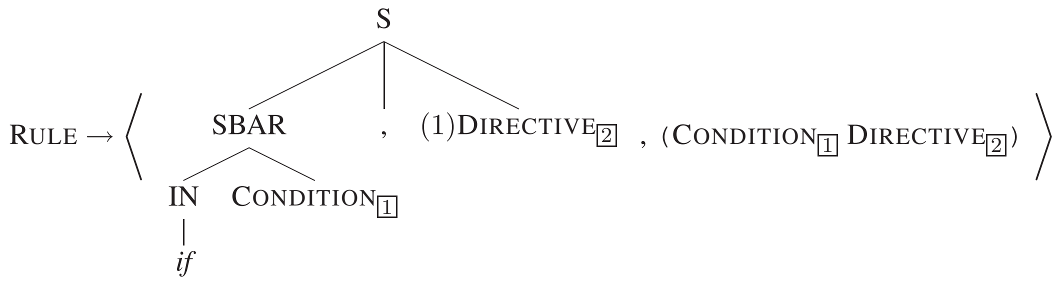
Since the meaning of a sentence clearly depends on how the words are combined, the semantic interpretation of a sentence must depend on the NL syntax. In Section 3.1, we showed that the parsing model of WASP is an SCFG that defines a semantic grammar for the NL stream. However, since an SCFG is induced without relying on any prior knowledge about the NL syntax, the resulting SCFG may not be linguistically-motivated, i.e. the NL substrings mapped by rules may not be what a linguist would call constituents.

If the NL syntax is known *a priori*, WASP should be made to take advantage of it, producing only parses that are consistent with the NL syntax. One way to obtain prior knowledge about the NL syntax is through syntactic annotations of the training set. In this section, we describe our initial work on *tree transformations*, in which a semantic parser accepts the syntactic parse of a sentence, rather than the sentence itself, as input, and transforms the syntactic parse into an MR using rules with tree patterns. Both the semantic parser and the syntactic parser from which syntactic parses are obtained are trained on fully-annotated syntactic parses. We will first define tree transformations, and then see if the extra syntactic annotations provide any useful constraints to the induction of a semantic parser.

4.1.1 Tree Transformations and Tree Patterns

Tree transformation is a topic traditionally studied in the context of logic and term rewriting systems (Gécseg & Steinby, 1997). Rounds (1970) motivates it as a mathematical model of transformational grammars (Chomsky, 1957). Shieber and Schabes (1990) introduces a class of tree transducers called synchronous tree-adjoining grammars (STAG) that characterize correspondences between natural languages. Knight and Graehl (2005) proposes using probabilistic tree transducers as a syntax-based alignment model.

Our tree-based parsing model is based on a tree transducer, which is similar to an SCFG as defined in Section 3.1, except that each pattern consists of a tree of terminal and non-terminal symbols. Recall that the term *non-terminal* always refers to non-terminal symbols of an MRL grammar. In a tree pattern, non-terminals can only be at the leaves of a tree. The following is a sample rule with a tree pattern:



Note that syntactic markers such as S and IN, like the word *if*, are terminal symbols. Each rewriting operation replaces a non-terminal in the NL stream with a fragment of a syntactic parse tree. Parsing is complete when all non-terminals have been rewritten, resulting in a complete syntactic parse tree. The symbol (1) preceding the DIRECTIVE non-terminal denotes a *node gap* of size 1, which means that at most one node can be skipped along the path from the node DIRECTIVE to its parent S. Since all non-terminals are at the leaves of a pattern, the tree transducer is *regular* (Rounds, 1970). It is equivalent to a synchronous tree-substitution grammar, a simplified version of STAG with substitution as the only composition operation (i.e. no adjunction).

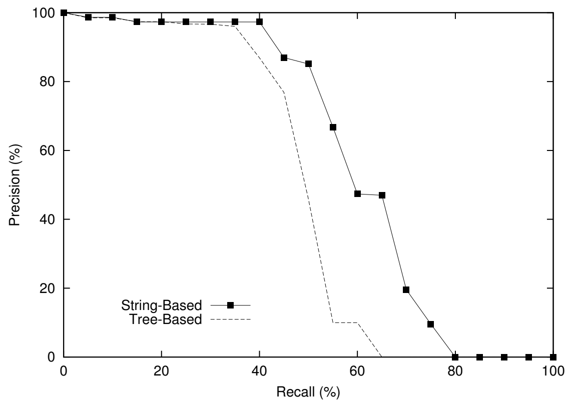
Word alignments are used for acquiring a tree-based lexicon. The rule extraction algorithm for tree patterns is a straightforward extension of the one for string patterns. In addition to words that are aligned, minimal projections of the aligned words are also included in a tree pattern. A *minimal projection* of a set of words is defined as the smallest constituent whose span covers all words in the set. If there is more than one minimal projection for a given set of words due to unary expansions, then all minimal projections are included in a tree pattern. Node gaps are inserted when the extracted nodes are not adjacent to each other in the original syntactic parse tree. An Earley parser similar to the one presented in Schabes and Shieber (1994) is used for parsing given a tree-based lexicon. It is also used for estimating parameters of a probabilistic model, which is the same as in Section 3.3.

4.1.2 Experiments and Discussion

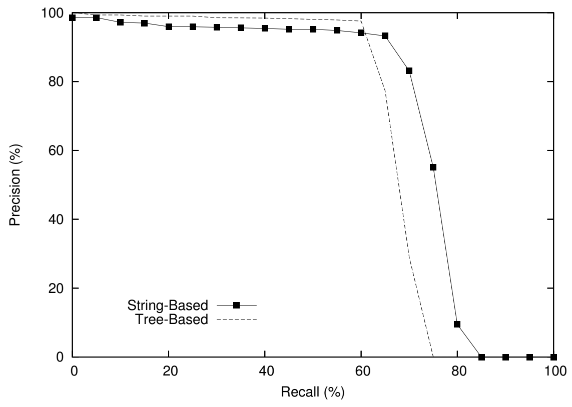
We evaluated tree-based rules by running experiments based on standard 10-fold cross validation. Details of the experiments were similar to Section 3.4. The only difference was that syntactic parses were needed for both training and testing in this case. Syntactic parses were obtained using Bikel's (2004) implementation of Collins' parser (Model 2) (Collins, 1997). The parser was trained using gold-standard syntactic parses of the training sentences, along with Sections 2–21 of the Wall Street Journal (WSJ) portion of the Penn Treebank. Gildea (2001) showed that combining two corpora in a single parsing model could improve parsing accuracy on either corpus, and we found that by adding a few domain-specific training sentences (~ 100) to the much bigger WSJ corpus (~ 40000), one could effectively bias the syntactic parser toward specific domains. Using the trained syntactic parser, the most likely syntactic parses for both training and test sentences were obtained. These parses were used for training and testing a semantic parser. In our experiments, we used a maximum-entropy model as the probabilistic model.

Figure 15 shows the precision of a semantic parser at various levels of recall. Recall level is varied by imposing a minimum threshold on the unnormalized parse probabilities given by a maximum-entropy model, $\exp \sum_i \lambda_i^\star f_i(\mathbf{d}^\star)$. In the GEOQUERY domain, precision is significantly higher for a tree-based parser when recall is 60% or below ($p = 0.032$ at 60% recall). On the other hand, in the ROBOCUP domain, precision remains the same. In both domains, a string-based parser can reach recall levels that a tree-based parser cannot reach, and the maximum recall attainable by a tree-based parser is 6–10% lower on average (not shown in the figure).

The reason for the low recall is twofold. First, due to the extra syntactic markers and bracketings, a tree pattern tends to be more specific than a string pattern. Second, while a string pattern can match any substring of a sentence, a tree pattern can only match a single constituent in a syntactic parse. Since not all substrings of a sentence form a constituent, the *span* of a tree pattern is often greater than a string pattern. This in turn causes more node merging during the rule extraction process (Section 3.2.1), leading to longer templates. In other words, derivations need to be shorter in order to maintain isomorphism of the NL and MR (semantic) parse trees. As a result, templates in a tree-based rule are on average 19–26% longer than in a string-based rule, which is a sign of overfitting. Note that generalization error of the syntactic parser is not a concern here, since there is no statistically significant difference in performance when gold-standard syntactic parses are used for training and testing. This seems to suggest that the cause of overfitting is not the NL syntax *per*



(a) ROBOCUP



(b) GEOQUERY

Figure 15: Precision-recall curves comparing string- and tree-based rules

se, but a mismatch between the NL and MRL syntax. For example, in CLANG, the meaning of the phrase *player 4 has the ball in our midfield* is represented by a conjunction of two conditions `bowner` (ball owner) and `bpos` (ball position). This MR is structurally different from the syntactic parse of the phrase, where *the ball* is an object of the verb *has*, and *in our midfield* is a prepositional phrase that in turn adjoins *the ball*. As a result, compositional transformation of the phrase is not possible at the condition level. In the string transformation case, however, a pattern can match non-constituents, so the non-constituent *player 4 has* can be transformed into a `bowner` condition without losing accuracy, because in CLANG a ball is the only thing a player can possess. Such structural mismatch between NL and MRL is particularly obvious for CLANG, because it is not designed with NL interpretation in mind. Note that this problem is not restricted to semantic parsing alone. Wu (1997) describes how it affects bilingual segmentations of two unrelated natural languages, Chinse and English, where a monolingually acceptable segmentation in Chinese may not agree with the words in English, and vice versa.

These results suggest that tree patterns can easily overfit, although they do improve precision in some cases. After all, tree transformation is not the only way to utilize syntactic information. In particular, features extracted from syntactic parses can be incorporated into the current maximum-entropy model (or any parametric models allowing feature overlap), assuming those features can be computed efficiently. Due to its flexibility, it is by far the most popular approach to exploiting syntactic information for machine translation (Och et al., 2003) and semantic role labeling (Carreras & Màrquez, 2005), and it should fit in the current semantic parsing framework seamlessly. The task is therefore to find a set of syntactic features that are discriminative and can be easily computed at the same time. We plan to experiment with this approach using features based on phrase boundaries, word dependencies, verb subcategorization frames, and so on.

4.2 Utilizing Additional Domain Knowledge

An important characteristic of WASP is its use of domain knowledge for guiding inductive learning. Domain knowledge comes in two forms:

1. A grammar of the target MRL that supplies the set of non-terminals on which generalizations are

based. It also defines the set of MRs that are syntactically well-formed.

2. Initial rules that provide default transformations for entities not encountered in the training data, such as numbers and place names.

In this section, we focus our attention on the MRL grammar. An MRL grammar determines the set of MR translations that are possible given an input sentence. For a well-designed MRL, all of these candidate MRs naturally convey meanings that are well-defined, but some MRs may be more plausible than others. To see this, consider the `loc_2` relation in the functional GEOQUERY language. It takes a finite set of places, X, and returns a finite set of places that are located in X. For example, the city of Austin would be in the set denoted by `loc_2(stateid('texas'))` (the set of places located in the state of Texas), and the University of Texas at Austin would be in the set denoted by `loc_2(cityid('austin','tx'))` (the set of places located in the city of Austin, Texas). Also consider the `city` and `state` relations, which take a finite set of places, X, and returns a subset of X that are cities and states, respectively. For example, the city of Austin would be in the set denoted by `city(loc_2(stateid('texas')))` (the set of cities located in the state of Texas). On the other hand, the expression `state(loc_2(cityid('austin','tx')))` would denote an empty set, for there are no states located in any U.S. cities. Therefore, `city(loc_2(state('texas')))` is much more plausible than `state(loc_2(cityid('austin','tx')))` in the GEOQUERY domain. Note that in domains other than U.S. geography, `state(loc_2(cityid(.)))` can be equally plausible, e.g. the state of Vatican City enclosed in the city of Rome, Italy.

Here we say an MR is *not plausible* if it is known that it cannot denote any actual entities in the application domain. Otherwise, the MR is plausible. This definition of plausibility allows us to talk about *states that border Alaska*, while ruling out meanings that are nonsensical, e.g. *the capital of a river*. Note that for certain application domains, an alternative definition of plausibility may be more suitable, e.g. ROBOCUP. For now, we will settle with this particular definition, which will be made more precise later for the GEOQUERY domain. It is desirable to have an MRL grammar where most syntactically well-formed MRs are plausible with respect to the application domain. An MRL grammar with such quality is said to be *tight*. A tight MRL grammar is desirable because less probability mass is allocated to MR translations that are not plausible. The reduced perplexity comes at the risk of excluding completely legitimate MRs from the parser output, so the exact definition of plausibility needs to be judiciously chosen for each individual application domain. This is where additional domain knowledge comes into play. Experience shows that a tight MRL grammar is necessary for good semantic parsing performance, and a loose (as opposed to tight) MRL grammar can cause the parsing algorithm to break down.

4.2.1 Constructing a Tight MRL Grammar

Now the problem is how to construct a tight grammar for the MRL. We first define the notion of plausibility for the GEOQUERY domain, then describe how it can be used for constructing a tight MRL grammar.

The notion of plausibility can be defined on a relation-to-relation basis. For example, it is found that with respect to the GEOQUERY domain, the following relations are plausible:

a city may be located in a state;
a lake may be located in a state;
a mountain may be located in a state;
a river may be located in a state;
a city may be located in a country;

a lake may be located in a country . . .

while the following relations are not:

a state may *not* be located in a city;
a state may *not* be located in a lake . . .

Whether a relation is plausible or not can be determined by examining the training data. If a relation appears in the training data, then it is assumed to be plausible. All relations in the transitive closure of plausible relations are assumed to be plausible. Other relations are not plausible. Each of these relations can be represented by a distinct production in an MRL grammar. A tight MRL grammar is constructed by restricting its productions to those that represent plausible relations. For example, a tight grammar for the functional GEOQUERY language (which is currently used in our experiments) would be as follows:

$$\begin{aligned}
\textsc{City} &\rightarrow \texttt{loc\_2(}\textsc{State}\texttt{)} \\
\textsc{Lake} &\rightarrow \texttt{loc\_2(}\textsc{State}\texttt{)} \\
\textsc{Mountain} &\rightarrow \texttt{loc\_2(}\textsc{State}\texttt{)} \\
\textsc{River} &\rightarrow \texttt{loc\_2(}\textsc{State}\texttt{)} \\
\textsc{City} &\rightarrow \texttt{loc\_2(}\textsc{Country}\texttt{)} \\
\textsc{Lake} &\rightarrow \texttt{loc\_2(}\textsc{Country}\texttt{)} \\
&\ldots
\end{aligned}$$

To see that this grammar is tight, consider `state(loc_2(cityid('austin','tx')))`, which would now be judged ungrammatical because STATE → `loc_2(`CITY`)` is not in the grammar. In other words, all syntactically well-formed MRs are assumed to be plausible, and vice versa. Reasonable semantic parsing performance is obtained by using this tight GEOQUERY grammar (Section 3.4.1), whereas a grammar that abstracts all places using a single PLACE non-terminal symbol would lower the precision of a semantic parser to about 50%.

The better precision comes at a cost, however. Now that the `loc_2` relation is specialized to handle arguments of different types, rules that are learned for one particular specialization are no longer applicable to other specializations. For example, in English, the preposition *in* indicates a `loc_2` relation regardless of argument types (e.g. *cities in this state*, *lakes in this country*). But the use of specialized relations would prevent such important generalizations from being made. Moreover, depending on the dimensionality of relations, there may be exponentially many specialized productions, with respect to the number of specialized non-terminals (e.g. CITY, LAKE) being introduced (as seen above). The proliferation of productions in the MRL grammar is bad because it increases the number of parameters that need to be estimated for both the word alignment model and the semantic parsing model, which in turn aggravates the problem of data sparsity.

A possible solution to this problem, which we have not implemented yet, is to group all specialized productions into equivalence classes. For example, specialized productions for the `loc_2` relation would belong to the same equivalence class. Such equivalence classes are used in the following:

1. Parameters for an equivalence class are *tied* in an alignment model, i.e. forced to have the same value.

2. Rules extracted for a specialized production are adapted for use in the entire equivalence class, by substituting a non-terminal for another. For example, if a rule $\textsc{City} \rightarrow \langle \textit{in}\ \textsc{State}_{\boxed{1}},\ \texttt{loc\_2(}\textsc{State}_{\boxed{1}}\texttt{)} \rangle$ is extracted, then it would be adapted to $\textsc{Lake} \rightarrow \langle \textit{in}\ \textsc{State}_{\boxed{1}},\ \texttt{loc\_2(}\textsc{State}_{\boxed{1}}\texttt{)} \rangle$, etc.

3. Parameters for rules that are adaptations of each other are tied in a semantic parsing model.

In effect, tying of parameters based on equivalence classes allows for a tight MRL grammar without increasing the complexity of a semantic parsing model. It allows generalizations of rules to be made, so a rule learned for one specialized production can be applied to other specializations in the same equivalence class. However, since the parameter tying scheme assumes a notion of plausibility based on specializations of relations, it is unclear whether this scheme can be carried over to other application domains.

4.2.2 Model-Theoretic Semantic Interpretation

Earlier we stated that an MR is not plausible if it is known that it cannot denote any actual entities in the application domain. A straightforward way to check the plausibility of an MR is then to examine the set of entities denoted by the MR. If the set is not empty, then the MR is plausible. Otherwise, the MR is not plausible. (Note that this is not entirely true, considering *states that border Alaska*, which does not denote any actual entities, but we will get back to this later.) This leads us to adopt a *model-theoretic semantics* of the MRL, where the interpretation of an MR is the set of entities that it denotes relative to a world model. In the GEOQUERY domain, the world model is the GEOQUERY database, and the model-theoretic interpretation of an MR is the set of entities that it denotes according to the GEOQUERY database. For example, the model-theoretic interpretation of `city(loc_2(stateid('texas')))` would be the set of all cities located in the state of Texas according to the database, i.e. $\{\text{AUSTIN}, \text{DALLAS}, \text{HOUSTON}, \ldots\}$, and the interpretation of `state(loc_2(cityid('austin','tx')))` would be an empty set. Given this model-theoretic semantics, we can disambiguate the meaning of a phrase or sentence by preferring non-empty interpretations over empty ones.

This model-theoretic semantics can be implemented by augmenting each transformation rule with a function that computes the denotation of an MR given the denotations of its sub-parts. For example, the rule PLACE $\rightarrow$ $\langle$*texas*, `stateid('texas')`$\rangle$ would be augmented by a constant, $\{\text{TEXAS}\}$. The rule PLACE $\rightarrow$ $\langle$*in* PLACE$_{\boxed{1}}$, `loc_2(`PLACE$_{\boxed{1}}$`)`$\rangle$ would be augmented by a function from $2^{\mathcal{E}}$ to $2^{\mathcal{E}}$ (where $\mathcal{E}$ is the set of entities in the world model) that maps $\{\text{TEXAS}\}$ to the set of all places located in the state of Texas. The rule PLACE $\rightarrow$ $\langle$*cities* PLACE$_{\boxed{1}}$, `city(`PLACE$_{\boxed{1}}$`)`$\rangle$ would be augmented by a function from $2^{\mathcal{E}}$ to $2^{\mathcal{E}}$ that maps the set of all places located in the state of Texas to the set of all *cities* located in the state of Texas. In general, each rule is augmented by a function from $(2^{\mathcal{E}})^k$ to $2^{\mathcal{E}}$, where k is the number of non-terminals in the rule template. Computation of denotations goes on until the denotation of a complete MR of a sentence is obtained. Schuler (2003) shows how denotations can be efficiently shared among possible derivations given an input sentence, such that the denotations of all possible derivations can be computed in polynomial time.

The main problem with this implementation is that the computation of denotation functions can be very costly, considering that each function evaluation amounts to a database query. This implementation can be simplified by reducing each denotation to a set of *entity types*, e.g. $\{\text{CITY}, \text{LAKE}, \text{MOUNTAIN}, \ldots\}$. Each transformation rule is then augmented by a function from $(2^E)^k$ to 2^E, where E is the set of entity types in the world model, and k is the number of non-terminals in the rule template. For example, the rule PLACE $\rightarrow$ $\langle$*texas*, `stateid('texas')`$\rangle$ would be augmented by a constant, $\{\text{STATE}\}$. The rule PLACE $\rightarrow$ $\langle$*in* PLACE$_{\boxed{1}}$, `loc_2(`PLACE$_{\boxed{1}}$`)`$\rangle$ would be augmented by a function from 2^E to 2^E that maps $\{\text{STATE}\}$ to $\{\text{CITY}, \text{LAKE}, \text{MOUNTAIN}, \text{RIVER}\}$. Evaluation of such denotation functions can be done by looking up a simple, pre-computed table. In addition, this simplified implementation has the advantage that phrases like *states that border Alaska* may have a non-empty denotation (e.g. $\{\text{STATE}\}$).

This model-theoretic semantics represents a divorce of syntactic well-formedness and plausibility, which allows a loose MRL grammar to be used without increasing perplexity of the semantic parsing model. More importantly, the set-theoretic semantic interpretations can be generalized to incorporate other types of truth conditions (i.e. conditions under which a sentence would be true), which can be useful when defining the notion of plausibility in other application domains. From a linguistic point of view, the model-theoretic framework is a shift from *interpretive semantics*, which argues that the sole function of the semantic component is to assign semantic interpretations to existing phrase markers (Katz & Postal, 1964). WASP as defined so far is clearly interpretive, because the output MR is always based on the most probable parse given an input sentence (Equation 9). By preferring certain interpretations over others based on model-theoretic semantics, we allow meanings to directly influence the parsing process. In future work, we hope to show that this paradigm shift indeed translates into better semantic parsing performance.

4.3 Syntax-Aware Word-Based Alignment Models

One crucial component of WASP is a word alignment model. In previous text, we explained the use of a word alignment model in lexical learning (Section 3.2), and the need for phrasal coherence in a word alignment (Section 3.2.2). In Section 2.3.2, we showed that to build an alignment model that strictly observes phrasal coherence (as are most syntax-based alignment models) requires transformation rules that model the reordering (and addition, deletion, etc.) of tree nodes. It follows that our SCFG-based parsing model is nothing more than a word alignment model that strictly observes phrasal coherence, and returns only syntactically well-formed MR translations when given an NL sentence as input. Since the space of all possible transformation rules is huge, it is necessary to have a simpler, word-based alignment model for bootstrapping the rule learning process. This is very much like the training of many phrase-based alignment models, which requires a simpler, word-based alignment model for the acquisition of a phrasal lexicon (Och et al., 1999; Tillmann, 2003; Venugopal et al., 2003), and the training of IBM Model 5, which requires the simpler Models 1–4 for initialization (Brown et al., 1993).

Being a first-order word-based alignment model, IBM Model 5 generally observes phrasal coherence (Section 2.3.1). Nevertheless, incoherent alignments do show up quite often, considering that it takes only one bad link to destroy an entire hierarchical structure of a sentence (Figure 8). In Section 3.2.2, we presented a simple greedy algorithm for removing links that destroy phrasal coherence. Although it is shown to be quite effective in the current domains (Section 3.4.2), it has the following shortcomings:

1. Productions whose links are removed are usually associated with some other words in a sentence. But the link removal algorithm makes no attempt to re-establish such associations other than adding links taken from a reverse alignment, which may not cover all cases.

2. If the link removal algorithm leaves a production with no links, then there are two possible outcomes: (1) the extracted rule would not be lexicalized; or (2) no rule would be extracted, until node merging causes a rule to be extracted for some production higher up in the MR parse tree. Either way the extracted rule would not be very accurate because of the missing links.

This motivates our search for a more principled way of promoting phrasal coherence in a word alignment model. Since phrasal coherence is defined with respect to an MRL grammar, the word alignment model should be made to exploit the MRL syntax, and hence be *syntax-aware*.

One problem with the current alignment model (IBM Model 5) is that while it is trained to maximize the likelihood of the training data, alignment quality is judged by a criterion which is quite different. Recall that in Section 3.2.2, we introduced a function $v(\mathbf{a})$ that computes the total number of violations of the isomorphism constraint in an alignment $\mathbf{a}$. When $v(\mathbf{a}) = 0$, rules can be extracted for all aligned productions in the linearized MR parse. When $v(\mathbf{a}) > 0$, some nodes in the linearized MR prase must be merged before rules can be extracted for them, and this is exactly the situation that we would like to avoid. In other words, an alignment, $\mathbf{a}$, is judged as good when $v(\mathbf{a})$ is small, and is judged as bad when $v(\mathbf{a})$ is large. Our goal is then to find an optimal alignment, $\mathbf{a}^\star$, for each training example such that $v(\mathbf{a}^\star)$ is minimized. This can be done in at least two ways:

1. Optimize the parameters of an alignment model with respect to alignment quality as measured by $v(\mathbf{a})$. This is closely related to minimum classification error rate training of speech recognizers (Juang et al., 1997), which is based on the minimization of a loss function that directly links to the evaluation metric by which performance of a recognizer is judged, namely classification error rate. Och (2003) uses a similar strategy for training a log-linear alignment model such that various evaluation metrics specific to machine translation (e.g. BLEU and NIST scores) are optimized.

2. Develop an optimal decision rule with respect to a loss function that directly links to $v(\mathbf{a})$, and use it for decoding an existing alignment model. This is called the minimum Bayes risk approach. Goodman (1996) uses this technique for finding syntactic parses that maximize labeled recall. Kumar and Byrne (2002) uses a loss function that incorporates various syntactic features of input sentences to obtain an optimal word alignment from a lattice of most likely alignments based on IBM Model 3. In this case, the training of an alignment model does not involve any knowledge about syntax, while the decoding of the resulting alignment model is syntactically-guided.

While the minimum Bayes risk approach is arguably weaker than the minimum error rate approach because alignments having the lowest $v(\mathbf{a})$ are not necessarily among the most likely ones, it tends to be more flexible since only modification of a decoder is involved. In either case, since the training or decoding criterion is directly associated with $v(\mathbf{a})$, which is defined with respect to an MRL grammar, the resulting algorithm for finding an optimal word alignment is syntax-aware. We intend to develop loss functions that lead to efficient and effective training and decoding algorithms for word-based alignment models, which are specific to our goal of promoting phrasal coherence in word alignments.

4.4 Toward More Complex Domains

We conclude Section 4 by discussing the possibility of extending WASP to application domains other than those currently considered. So far, the application domains being considered are restricted domains with a controlled vocabulary. Since both word alignment models and SCFGs have been successfully applied to machine translation of newswire text and parliament proceedings (Brown et al., 1993; Och & Ney, 2003; Yamada & Knight, 2001; Chiang, 2005), we expect that WASP should scale to larger vocabularies as well. In particular, while CFG has been shown to be inadequate for modeling non-local phenomena in natural languages (Shieber, 1985), leading to the development of *mildly context-sensitive grammars* such as tree-adjoining grammars (TAG) (Joshi, 1985) and combinatory categorial grammars (CCG) (Steedman, 2000), substantial mileage can be achieved by using CFG to model natural languages, including certain non-local phenomena such as wh-movements (Collins, 1997). However, various basic assumptions that WASP makes may hinder its ability to tackle application domains other than the ones being considered in this work. In the following sections, we examine two of these basic assumptions, functional target MRLs and clean training data, and see if these assumptions pose a problem at all, and if so, how WASP can be extended to eliminate these assumptions. Note that this would constitute a future research plan for the longer term, since corpus annotation (or acquistion and adaptation of existing ones) is required for the development of semantic parsers for additional application domains.

4.4.1 Alternative Meaning Representation Languages

In this work, we assume the use of a functional language as the target MRL. Both domains on which WASP is tested use an MRL that is variable-free, unlike most MRLs one would encounter in computational semantics that are based on predicate logic (Zelle & Mooney, 1996; Blackburn & Bos, 2005). A natural question to ask is how expressive these functional languages are. It turns out that these functional languages are merely shorthand for a subset of logical languages, so the real question is whether this subset is adequate for interpretation of natural languages.

To see how functional languages are shorthand for a subset of logical languages, consider the following rules based on the functional GEOQUERY language:

$$\begin{array}{rl}
\textsc{State} \rightarrow & \langle \textit{missouri}, \texttt{stateid('missouri')} \rangle \\
\textsc{Place} \rightarrow & \langle \textit{in}\ \textsc{State}_{\boxed{1}}, \texttt{loc\_2}\,(\textsc{State}_{\boxed{1}}) \rangle \\
\textsc{River} \rightarrow & \langle \textit{rivers}\ (2)\ \textsc{Place}_{\boxed{1}}, \texttt{river}\,(\textsc{Place}_{\boxed{1}}) \rangle \\
\textsc{Num} \rightarrow & \langle \textit{how many}\ \textsc{River}_{\boxed{1}}, \texttt{count}\,(\textsc{River}_{\boxed{1}}) \rangle \\
\textsc{Query} \rightarrow & \langle \textsc{Num}_{\boxed{1}}\ \textit{?}, \texttt{answer}\,(\textsc{Num}_{\boxed{1}}) \rangle
\end{array} \tag{23}$$

These rules can be rewritten in the form of lambda functions, which generate logical forms instead of functional forms by replacing function applications with variable assignments:

$$\begin{array}{rl}
\textsc{State} \rightarrow & \langle \textit{missouri}, \lambda x.x = \textsc{stateid}(\textsc{missouri}) \rangle \\
\textsc{Place} \rightarrow & \langle \textit{in}\ \textsc{State}, \lambda p.\lambda x.\exists y\ \textsc{loc}(x,y) \wedge p(y) \rangle \\
\textsc{River} \rightarrow & \langle \textit{rivers}\ (2)\ \textsc{Place}, \lambda p.\lambda x.\textsc{river}(x) \wedge p(x) \rangle \\
\textsc{Num} \rightarrow & \langle \textit{how many}\ \textsc{River}, \lambda p.\lambda x.\exists y\ \textsc{count}(y, p(y), x) \rangle \\
\textsc{Query} \rightarrow & \langle \textsc{Num}\ \textit{?}, \lambda p.\exists x\ \textsc{answer}(x, p(x)) \rangle
\end{array} \tag{24}$$

The rules in (23) and (24) would transform the NL question *How many rivers are there in Missouri?* into the following functional and logical forms, respectively:

$$\texttt{answer(count(river(loc\_2(stateid('missouri')))))}$$
$$\exists x\ \textsc{answer}(x, (\exists y\ \textsc{count}(y, (\exists z\ \textsc{river}(y) \wedge \textsc{loc}(y,z) \wedge z = \textsc{stateid}(\textsc{missouri})), x)))$$

The functional form is equivalent to the logical form, which is in the original GEOQUERY language. In fact, each function in the functional GEOQUERY language is defined by a lambda function that generates the original GEOQUERY language. These lambda functions restrict the possibilities of variable binding, such that variables will have a limited scope. For example, the scope of z in the logical form above is restricted to the formula $\textsc{river}(y) \wedge \textsc{loc}(y,z) \wedge z = \textsc{stateid}(\textsc{missouri})$. While such limited scoping of variables is prevalent in the GEOQUERY corpus, it critically fails to account for an important linguistic phenomenon: sentences whose meanings are dependent on earlier discourse (e.g. the ATIS domain as described in Section 2.1.1). Such context-dependent sentences will be the topic of the rest of this section.

Anaphora Resolution and Discourse Representation Structures In the context of NL interfaces, sentences whose meanings are dependent on earlier discourse usually appear in the form of follow-up questions in a dialog. A follow-up question is a question that refers back to an entity that has been mentioned earlier in a discourse. For example, in the following dialog in the GEOQUERY domain:

USER: *What is the capital of Texas?*
SYSTEM: *Austin.*
USER: *How many people live in the city?*
SYSTEM: *656,562 as of April 1, 2000.*

The second question is a follow-up question, where the phrase *the city* is an anaphoric expression that refers to the capital of Texas (the antecedant), mentioned in the first question. To answer the second question, the computer must keep a record of entities that has been mentioned so far, and resolve the anaphoric expression based on the structure of discourse and various syntactic and semantic constraints.

Discourse Representation Theory (DRT) (Kamp & Reyle, 1993) is an approach to semantics that formalizes this record-keeping idea. According to DRT, each discourse has a main *discourse representation structure* (DRS), which can be seen as a mental model constructed during the process of discourse comprehension. Each DRS consists of a set of *discourse referents*, which can be seen as variables, and a set of conditions specifying the properties of the discourse referents and the way they are related to each other. As a processor receives a sentence fragment, the processor parses it and fills in the main DRS by introducing new discourse referents and adding new conditions. For example, the phrase *the capital of Texas* would cause the processor to add a new discourse referent, x. Conditions would be added to ensure that the discourse referent is the capital of Texas. Later, when the phrase *the city* is processed, another discourse referent, y, would be added. Most sigificantly, the processor would identify it with the discourse referent for the capital of Texas introduced earlier, by adding an equality condition, $y = x$. The *meaning* of a sentence fragment is therefore defined as its potential to change the mental model during a discourse, such that subsequent anaphoric expressions can be properly resolved based on the mental model.

It turns out that the meaning of a sentence fragment itself can be expressed in a DRS, which is then *merged* with the main DRS associated with a discourse to obtain an updated mental model. Similarly, the meaning of a sentence can be expressed in a DRS, which is constructed by merging the meanings of its sub-parts expressed in DRSs, in much the same way a logical formula is constructed through applications of lambda functions (Blackburn & Bos, 2005). This so-called λ-DRT approach of constructing the meaning of a sentence by combining the meanings of its sub-parts (Muskens, 1996) suggests an alternative MRL consisting of DRSs which allows us to deal with anaphoric expressions using our SCFG-based framework. In particular, we are interested in reducing DRSs to a variable-free, functional form. Shan (2001) proposes a way this can be done based on the theory of variable-free semantics (Jacobson, 1999). We hope to show that such alternative MRLs based on DRT can extend the current semantic parsing framework to the follow-up question scenario.

4.4.2 Learning from Noisy Training Data

Currently, WASP learns only from positive examples. This means that a *parallel corpus* is used as the training data, where all NL sentences are assumed to be meaningful, and each sentence is paired up with an MR that correctly conveys the meaning of the sentence. Unfortunately, such parallel corpora may not be available for some scenarios. For example, consider the ROBOCUP commentary task, where reports of a simulated soccer game are interpreted on the basis of visual data captured in the form of game logs (André et al., 2000). Before learning a semantic parser that interprets soccer commentaries, abstract symbolic descriptions are extracted from the low-level visual data (Siskind, 2001; Gorniak & Roy, 2004). However, since such symbolic descriptions cover all events happening on the entire field, and each sentence in a commentary necessarily refers to only part of these events, we are faced with the problem of *referential ambiguity*, where the exact portion of symbolic descriptions that a sentence refers to has yet to be determined. This is a case of 1-to-n

noise in the training data, where for each sentence, all but one candidate MR is irrelevant. Such noise can be generalized to the n-to-n case where not all NL utterances are relevant, e.g. when a soccer commentary diverges to a biographical sketch, irrelevant to events happening on a field.

These scenarios give rise to a *non-parallel corpus* (or a *comparable corpus*), where NL sentences and MRs are not sentence-aligned, and only part of them are bilingual translations of the same concepts. The task of finding correspondences between NL sentences and MRs is then a problem of *sentence alignment*, treating MRs as *sentences* in an MRL (Brown et al., 1991; Gale & Church, 1993). Note that unlike the word alignment task (Section 2.3), the sentence alignment task for comparable corpora does not assume a full translation of text, as the input articles are often aligned in topics but not in content. A wide variety of methods have been proposed for mining parallel sentences from such corpora (Manning & Schütze, 1999). Here we focus on methods that use lexical information to guide the alignment process (Kay & Röscheisen, 1993; Chen, 1993; Zhao & Vogel, 2002; Munteanu et al., 2004; Fung & Cheung, 2004). The intuition is that sentences that contain more words that are translations of each other tend to be parallel sentences, and conversely, words contained in sentences that are mutual translations tend to correspond to each other. Following Fung and Cheung's (2004) EM-based model, WASP can be adapted to use a non-parallel training corpus in the following manner:

1. Assuming that all bilingual sentence pairs (treating MRs as sentences) are equally likely to be mutual translations, learn transformation rules that map NL substrings to MRs using WASP.
2. Obtain an *alignment score* for each bilingual sentence pair $\langle \mathbf{e}, \mathbf{f} \rangle$, e.g. based on the joint probability of generating $\mathbf{e}$ and $\mathbf{f}$.
3. Update the probability for each bilingual sentence pair to be mutual translations, using alignment scores obtained in Step 2.
4. Learn new transformation rules based on the updated probability distribution in Step 3. Repeat Steps 2–4 until the resulting semantic parsing model converges.

The hidden variables of this EM-based algorithm are therefore the bilingual sentence pairs that are mutual translations. This algorithm crucially depends on a semantic parsing model (and thus a word alignment model) that is sensitive to the probability that a training example is correct (Steps 1 and 4). To our knowledge, none of the existing word alignment models take this probability distribution into account. Most current methods of sentence alignment circumvent this problem by assuming that a sentence pair is either mutual translations or not, with no uncertainty in between (Kay & Röscheisen, 1993; Munteanu et al., 2004; Fung & Cheung, 2004). This allows them to train word alignment models on only sentences that are deemed parallel, a Viterbi approximation of the more general EM-based framework presented above. We conjecture that the more general EM-based framework is more robust to variations in the degree of lexical overlap among parallel sentences, and thus more effective in finding them in a non-parallel corpus.

Nevertheless, based on prior knowledge, better estimates of the hidden variables of the EM algorithm can be obtained. Such prior knowledge plays an important role in most existing sentence alignment algorithms. For example, both Kay and Röscheisen (1993) and Zhao and Vogel (2002) assume roughly similar sentence order for the two languages under consideration, and Munteanu et al. (2004) constructs a non-parallel corpus of Arabic and English news stories by matching their publication dates. We expect that similar constraints can be devised for the task of semantic parsing, e.g. based on timestamps of utterances and events in the ROBOCUP commentary domain, which is especially important given that the signal-to-noise ratio can be very small in the MRL stream (compared to that in the NL stream).

In future, we plan to develop a version of WASP that accepts noisy training data. Apart from being more realistic about the context in which language acquisition occurs (Pinker, 1995), it is also a step toward robust natural language understanding in more complex domains, such as conversational interfaces (Zue & Glass, 2000) and multi-modal systems (André, 2003).

5 Conclusion

We presented a statistical approach to semantic parsing based on the synchronous context-free grammar. A statistical word alignment model is used for lexical acquisition. Initial evaluation on several real-world data sets showed that the algorithm performs favorably compared to existing learning methods, regardless of task complexity and word order. In future work, we plan to exploit extra knowledge about the NL syntax and the application domain. We also intend to construct a syntax-aware, word-based alignment model for better lexical acquisition. Finally, we plan to extend the algorithm to application domains in which the meanings of sentences may be context-dependent, and in which training data is noisy.

References

Aho, A. V., & Ullman, J. D. (1969a). Properties of syntax directed translations. *Journal of Computer and System Sciences*, *3*(3), 319–334.

Aho, A. V., & Ullman, J. D. (1969b). Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, *3*(1), 37–56.

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

Al-Onaizan, Y., Curin, J., Jahr, M., Knight, K., Lafferty, J., Melamed, I. D., Och, F. J., Purdy, D., Smith, N. A., & Yarowsky, D. (1999). Statistical machine translation. Tech. rep., The Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD.

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

André, E. (2003). Natural language in multimedia/multimodal systems. In Mitkov, R. (Ed.), *Handbook of Computational Linguistics*, pp. 650–669. Oxford University Press.

André, E., Binsted, K., Tanaka-Ishii, K., Luke, S., Herzog, G., & Rist, T. (2000). Three RoboCup simulation league commentator systems. *AI Magazine*, *21*(1), 57–66.

Androutsopoulos, I., Ritchie, G. D., & Thanisch, P. (1995). Natural language interfaces to databases: An introduction. *Journal of Natural Language Engineering*, *1*(1), 29–81.

Baker, J. K. (1979). Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pp. 547–550, Boston, MA.

Bellegarda, J. R., & Silverman, K. E. A. (2003). Natural language spoken interface control using data-driven semantic inference. *IEEE Transactions on Speech and Audio Processing*, *11*(3), 267–277.

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.

Booth, T. L., & Thompson, R. A. (1973). Applying probability measures to abstract languages. *IEEE Transactions on Computers*, *22*(5), 442–450.

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

Brown, P. F., Della Pietra, V. J., Della Pietra, S. A., & Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, *19*(2), 263–312.

Brown, P. F., Lai, J. C., & Mercer, R. L. (1991). Aligning sentences in parallel corpora. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL-91)*, pp. 169–176, Berkeley, CA.

Byrd, R. H., Nocedal, J., & Schnabel, R. B. (1994). Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, *63*(2), 129–156.

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

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

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. (1993). Aligning sentences in bilingual corpora using lexical information. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL-93)*, pp. 9–16.

Chen, S. F. (1995). Bayesian grammar induction for language modeling. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, pp. 228–235.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pp. 263–270, Ann Arbor, MI.

Chieu, H. L., & Ng, H. T. (2003). Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Computational Natural Language Learning (CoNLL-2003)*, pp. 160–163, Edmonton, Canada.

Chomsky, N. (1957). *Syntactic Structures*. Mouton & Co., The Hague.

Clark, A. (2001). Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the Fifth Conference on Computational Natural Language Learning (CoNLL-2001)*, Toulouse, France.

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

Collins, M., & Roark, B. (2004). Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pp. 111–118, Barcelona, Spain.

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

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.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*, 1–38.

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

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

Friedland, N. S., Allen, P. G., Matthews, G., Witbrock, M., Baxter, D., Curtis, J., Shepard, B., Miraglia, P., Angele, J., Staab, S., Moench, E., Oppermann, H., Wenke, D., Israel, D., Chaudhri, V., Porter, B., Barker, K., Fan, J., Chaw, S. Y., Yeh, P., Tecuci, D., & Clark, P. (2004). Project Halo: Towards a digital Aristotle. *AI Magazine*, *25*(4), 29–47.

Fung, P., & Cheung, P. (2004). Mining very-non-parallel corpora: Parallel sentence and lexicon extraction via bootstrapping and EM. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, pp. 57–63, Barcelona, Spain.

Gale, W. A., & Church, K. W. (1993). A program for aligning sentences in bilingual corpora. *Computational Linguistics*, *19*, 75–102.

Ge, R., & Mooney, R. J. (2005). A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pp. 9–16, Ann Arbor, MI.

Gécseg, F., & Steinby, M. (1997). Tree languages. In Rozenberg, G., & Salomaa, A. (Eds.), *Handbook of Formal Languages*, Vol. 3, pp. 1–68. Springer Verlag, Berlin.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*, 721–742.

Geman, S., & Johnson, M. (2002). Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proceedings 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 *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-01)*, Pittsburgh, PA.

Goodman, J. (1996). Parsing algorithms and metrics. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL-96)*, pp. 177–183, Santa Cruz, CA.

Gorniak, P., & Roy, D. (2004). Grounded semantic composition for visual scenes. *Journal of Artificial Intelligence Research*, *21*, 429–470.

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

Hendrix, G. G., Sacerdoti, E., Sagalowicz, D., & Slocum, J. (1978). Developing a natural language interface to complex data. *ACM Transactions on Database Systems*, *3*(2), 105–147.

Jacobson, P. (1999). Towards a variable-free semantics. *Linguistics and Philosophy*, *22*, 117–184.

Jelinek, F. (1985). Markov source modeling of text generation. In Skwirzinski, J. K. (Ed.), *The Impact of Processing Techniques on Communications*. Dordrecht, Nijhoff.

Jelinek, F. (1998). *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, MA.

Jelinek, F., & Lafferty, J. D. (1991). Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, *17*(3), 315–323.

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

Joshi, A. K. (1985). Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions. In Dowty, D. R., Karttunen, L., & Zwicky, A. (Eds.), *Natural Language Parsing*. Cambridge University Press, New York.

Juang, B., Chou, W., & Lee, C. (1997). Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Audio Processing*, *5*(3), 257–265.

Kamp, H., & Reyle, U. (1993). *From Discourse to Logic*. Kluwer, Dordrecht.

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

Katz, J. J., & Postal, P. M. (1964). *An Integrated Theory of Linguistic Descriptions*. MIT Press, Cambridge, MA.

Kay, M., & Röscheisen, M. (1993). Text-translation alignment. *Computational Linguistics*, *19*(1), 121–142.

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

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

Knoke, D., & Burke, P. J. (1980). *Log-Linear Models*. Sage Publications, Inc., Newberry Park, CA.

Koomen, P., Punyakanok, V., Roth, D., & Yih, W. (2005). Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pp. 181–184, Ann Arbor, MI.

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

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.

Kumar, S., & Byrne, W. (2002). Minimum Bayes-risk word alignments of bilingual texts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*, pp. 140–147, Philadelphia, PA.

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

Lari, K., & Young, S. J. (1990). The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, *4*, 35–56.

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

Macherey, K., Och, F. J., & Ney, H. (2001). Natural language understanding using statistical machine translation. In *Proceedings of the 7th European Conference on Speech Communication and Technology (EuroSpeech-01)*, pp. 2205–2208, Aalborg, Denmark.

Magerman, D. M. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, pp. 276–283, Cambridge, MA.

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

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

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

Melamed, I. D. (2004). Statistical machine translation by parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pp. 653–660, Barcelona, Spain.

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

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

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.

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

Muskens, R. (1996). Combining Montague semantics and discourse representation. *Linguistics and Philosophy*, *19*, 143–186.

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

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pp. 160–167, Sapporo, Japan.

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. (2003). Syntax for statistical machine translation. Tech. rep., The Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD.

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

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

Pereira, F. C. N., & Shabes, Y. (1992). Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL-92)*, pp. 128–135, Newark, Delaware.

Pinker, S. (1995). Language acquisition. In Gleitman, L. R., & Liberman, M. (Eds.), *Language* (2nd edition)., Vol. 1 of *An Invitation to Cognitive Science*, pp. 135–182. MIT Press, Cambridge, MA.

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 *Proceedings of the Twentieth International Conference on Computational Linguistics (COLING-04)*, Geneva, Switzerland.

Popescu, A.-M., Etzioni, O., & Kautz, H. (2003). Towards a theory of natural language interfaces to databases. In *Proceedings of the 2003 International Conference on Intelligent User Interfaces (IUI-2003)*, pp. 149–157, Miami, FL. ACM.

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

Quirk, C., Menezes, A., & Cherry (2005). Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pp. 271–279, Ann Arbor, MI.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

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

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

Riezler, S. (1998). *Probabilistic Constraint Logic Programming*. Ph.D. thesis, Universität Tübingen, Germany.

Riezler, S., Prescher, D., Kuhn, J., & Johnson, M. (2000). Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, pp. 480–487, Hong Kong.

Rounds, W. C. (1970). Mappings and grammars on trees. *Mathematical Systems Theory*, *4*(3), 257–287.

Schabes, Y., & Shieber, S. M. (1994). An alternative conception of tree-adjoining derivation. *Computational Linguistics*, *20*(1), 91–124.

Schuler, W. (2003). Using model-theoretic semantic interpretation to guide statistical parsing and word recognition in a spoken language interface. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pp. 529–536.

Shan, C. (2001). A variable-free dynamic semantics. In *Proceedings of the 13th Amsterdam Colloquium*, pp. 204–209, University of Amsterdam.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, *8*, 333–343.

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

Simmons, R., Goldberg, D., Goode, A., Montemerlo, M., Roy, N., Sellner, B., Urmson, C., Schultz, A., Abramson, M., Adams, W., Atrash, A., Bugajska, M., Coblenz, M., MacMahon, M., Perzanowski, D., Horswill, I., Zubek, R., Kortenkamp, D., Wolfe, B., Milam, T., & Maxwell, B. (2003). GRACE: An autonomous robot for the AAAI robot challenge. *AI Magazine*, *24*(2), 51–72.

Siskind, J. M. (2001). Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of Artificial Intelligence Research*, *15*, 31–90.

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

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

Stolcke, A., & Omohundro, S. M. (1994). Inducing probabilistic grammars by bayesian model merging. In Carrasco, R. C., & Oncina, J. (Eds.), *Grammatical Inference and Applications*, pp. 106–118. Springer.

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

Tang, L. R., & Mooney, R. J. (2000). Automated construction of database interfaces: Integrating statistical and relational learning for semantic parsing. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora(EMNLP/VLC-2000)*, pp. 133–141, Hong Kong.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. In *Proceedings of 18th Conference on Uncertainty in Artificial Intelligence (UAI-2002)*, pp. 485–492, Edmonton, Canada.

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

Tillmann, C. (2003). A projection extension algorithm for statistical machine translation. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP-03)*, pp. 1–8, Sapporo, Japan.

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

Venugopal, A., Vogel, S., & Waibel, A. (2003). Effective phrase translation extraction from alignment models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pp. 319–326, Sapporo, Japan.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, *13*(2), 260–269.

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

Yamada, K., & Knight, K. (2001). A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001)*, pp. 523–530, Toulouse, France.

Yamada, K., & Knight, K. (2002). A decoder for syntax-based MT. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 303–310, Philadelphia, PA.

Zelle, J. M., & Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 1050–1055, Portland, OR.

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

Zhao, B., & Vogel, S. (2002). Adaptive parallel sentences mining from web bilingual news collection. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM-2002)*, pp. 745–750.

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