Modern Natural Language Interfaces to Databases: Composing Statistical Parsing with Semantic Tractability

Ana-Maria Popescu Alex Armanasu Oren Etzioni David Ko Alexander Yates
University of Washington
{amp, alexarm, etzioni, daveko, ayates}@cs.washington.edu

Abstract

Natural Language Interfaces to Databases (NLIs) can benefit from the advances in statistical parsing over the last fifteen years or so. However, statistical parsers require training on a massive, labeled corpus, and manually creating such a corpus for each database is prohibitively expensive. To address this quandary, this paper reports on the PRECISE NLI, which uses a statistical parser as a "plug in". The paper shows how a strong semantic model coupled with "light re-training" enables PRECISE to overcome parser errors, and correctly map from parsed questions to the corresponding SQL queries. We discuss the issues in using statistical parsers to build database-independent NLIs, and report on experimental results with the benchmark ATIS data set where PRECISE achieves 94% accuracy.

1 Introduction and Motivation

Over the last fifteen years or so, much of the NLP community has focused on the use of statistical and machine learning techniques to solve a wide range of problems in parsing, machine translation, and more. Yet, classical problems such as building Natural Language Interfaces to Databases (NLIs) (Grosz et al., 1987) are far from solved.

There are many reasons for the limited success of past NLI efforts (Androutsopoulos et al., 1995). We highlight several problems that are remedied by our approach. First, manually authoring and tuning a semantic grammar for each new database is brittle and prohibitively expensive. In response, we have implemented a "transportable" NLI that aims to minimize manual, database-specific configuration. Second, NLI systems built in the 70s and 80s had limited syntactic parsing capabilities. Thus, we have an opportunity to incorporate the important advances made by statistical parsers over the last two decades in an NLI.

However, attempting to use a statistical parser in a database-independent NLI leads to a quandary. On the one hand, to parse questions posed to a *particular* database, the parser has to be trained on a corpus of questions specific to that database. Otherwise, many of the parser's decisions will be incorrect. For example, the Charniak parser (trained on the 40,000 sentences in the WSJ portion of the Penn Treebank) treats 'list' as a noun, but in the context of the ATIS database it is a verb.[1] On the other hand, manually creating and labeling a massive corpus of questions for each database is prohibitively expensive.

We consider two methods of resolving this quandary and assess their performance individually and in concert on the ATIS data set. First, we use a strong semantic model to correct parsing errors. We introduce a theoretical framework for discriminating between *Semantically Tractable* (ST) questions and difficult ones, and we show that ST questions are prevalent in the well-studied ATIS data set (Price, 1990). Thus, we show that the semantic component of the NLI task can be surprisingly easy and can be used to compensate for syntactic parsing errors. Second, we re-train the parser using a relatively small set of 150 questions, where each word is labeled by its part-of-speech tag.

To demonstrate how these methods work in practice, we sketch the fully-implemented PRECISE NLI, where a parser is a modular "plug in". This modularity enables PRECISE to leverage continuing advances in parsing technology over time by plugging in improved parsers as they become available.

The remainder of this paper is organized as follows. We describe PRECISE in Section 2, sketch our theory in Section 3, and report on our experiments in Section 4. We consider related work in Section 5, and conclude in Section 6.

2 The PRECISE System Overview

Our recent paper (Popescu et al., 2003) introduced the PRECISE architecture and its core algorithm for

[1] This is an instance of a well known machine learning principle — typically, a learning algorithm is effective when its test examples are drawn from roughly the same distribution as its training examples.

reducing semantic interpretation to a graph matching problem that is solved by MaxFlow. In this section we provide a brief overview of PRECISE, focusing on the components necessary to understanding its performance on the ATIS data set in Section 4.

To discuss PRECISE further, we must first introduce some terminology. We say that a database is made up of three types of *elements*: *relations*, *attributes* and *values*. Each element is unique: an *attribute* element is a particular column in a particular *relation* and each *value* element is the value of a particular *attribute*. A value is *compatible* with its attribute and also with the relation containing this attribute. An attribute is *compatible* with its relation. Each attribute in the database has associated with it a special value, which we call a *wh-value*, that corresponds to a wh-word (what, where, *etc.*).

We define a *lexicon* as a tuple (T, E, M), where T is a set of strings, called *tokens* (intuitively, tokens are strings of one or more words, like "New York"); E is a set of database elements, wh-values, and join paths; [2] and M is a subset of $T \times E$ — a binary relation between tokens and database elements.

PRECISE takes as input a lexicon and a parser. Then, given an English question, PRECISE maps it to one (or more) corresponding SQL queries. We concisely review how PRECISE works through a simple example. Consider the following question q: "What are the flights from Boston to Chicago?" First, the *parser plug-in* automatically derives a dependency analysis for q from q's parse tree, represented by the following compact *syntactic logical form*: $LF(q) = what(0), is(0,1), flight(1), from(1,2), boston(2), to(1,3), chicago(3)$.
$LF(q)$ contains a predicate for each question word. Head nouns correspond to unary predicates whose arguments are constant identifiers.

Dependencies are encoded by equality constraints between arguments to different predicates. The first type of dependency is represented by noun and adjective pre-modifiers corresponding to unary predicates whose arguments are the identifiers for the respective modified head nouns. A second type of dependency is represented by noun postmodifiers and mediated by prepositions (in the above example, "from" and "to"). The prepositions correspond to binary predicates whose arguments specify the attached noun phrases. For instance, "from" attaches "flight" to "boston". Finally, subject/predicate, predicate/direct object and predicate/indirect object dependency information is computed for the various verbs present in the question. Verbs correspond to binary or tertiary predicates whose arguments indicate what noun phrases play the subject and object roles. In our example, the verb "is" mediates the dependency between "what" and "flight". [3]

PRECISE's lexicon is generated by *automatically* extracting value, attribute, and relation names from the database. We manually augmented the lexicon with relevant synonyms, prepositions, *etc.*.

The *tokenizer* produces a single complete tokenization of this question and lemmatizes the tokens: `(what, is, flight, from, boston, to, chicago)`. By looking up the tokens in the lexicon, PRECISE efficiently retrieves the set of potentially matching database elements for every token. In this case, `what`, `boston` and `chicago` are value tokens, `to` and `from` are attribute tokens and `flight` is a relation token.

In addition to this information, the lexicon also contains a set of *restrictions* for tokens that are prepositions or verbs. The restrictions specify the database elements that are allowed to match to the arguments of the respective preposition or verb. For example, `from` can take as arguments a **flight** and a **city**. The restrictions also specify the join paths connecting these relations/attributes. The syntactic logical form is used to retrieve the relevant set of restrictions for a given question.

The *matcher* takes as input the information described above and reduces the problem of satisfying the semantic constraints imposed by the definition of a valid interpretation to a graph matching problem (Popescu et al., 2003). In order for each attribute token to match a value token, `Boston` and `Chicago` map to the respective values of the database attribute **city.cityName**, `from` maps to **flight.fromAirport** or **fare.fromAirport** and `to` maps to **flight.toAirport** or **fare.toAirport**. The restrictions validate the output of the matcher and are then used in combination with the syntactic information to narrow down even further the possible interpretations for each token by enforcing local dependencies. For example, the syntactic information tells us that "from" refers to "flight" and since "flight" uniquely maps to **flight**, this means that `from` will map to **flight.fromAirport** rather than **fare.fromAirport** (similarly, `to` maps to **flight.toAirport** and `what` maps to **flight.flightId**). Finally, the matcher compiles a list of all relations satisfying all the clauses in the syntactic logical form using each constant and narrows down the set

[2] A join path is a set of equality constraints between the attributes of two or more tables. See Section 3 for more details and a formal definition.

[3] PRECISE uses a larger set of constraints on dependency relations, but for brevity, we focus on those relevant to our examples.

of possible interpretations for each token accordingly. Each set of (constant, corresponding database element) pairs represents a *semantic logical form*.

The *query generator* takes each semantic logical form and uses the join path information available in the restrictions to form the final SQL queries corresponding to each semantic interpretation.

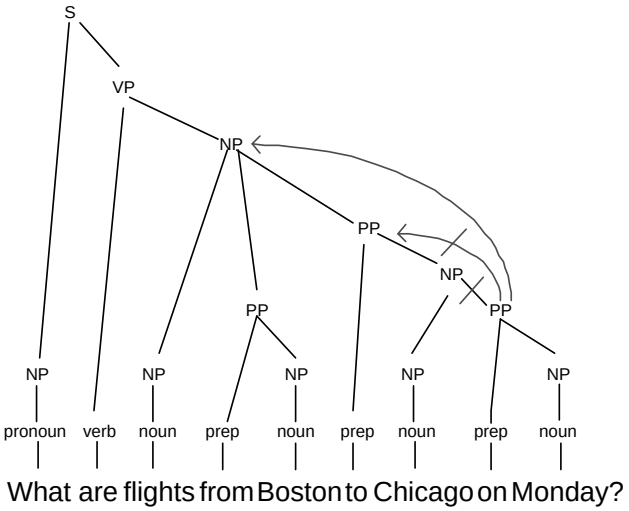

Figure 1: **Example of an erroneous parse tree corrected by PRECISE's semantic over-rides. PRECISE detects that the parser attached the PP "on Monday" to "Chicago" in error. PRECISE attempts to re-attach "on Monday" first to the PP "to Chicago", and then to the NP "flights from Boston to Chicago", where it belongs.**

2.1 Parser Enhancements

We used the Charniak parser (Charniak, 2000) for the experiments reported in this paper. We found that the Charniak parser, which was trained on the WSJ corpus, yielded numerous syntactic errors. Our first step was to hand tag a set of 150 questions with Part Of Speech (POS) tags, and re-train the parser's POS tagger. As a result, the probabilities associated with certain tags changed dramatically. For example, initially, 'list' was consistently tagged as a noun, but after re-training it was consistently labeled as a verb. This change occurs because, in the ATIS domain, 'list' typically occurs in imperative sentences, such as "List all flights."

Focusing exclusively on the tagger drastically reduced the amount of data necessary for re-training. Whereas the Charniak parser was originally trained on close to 40,000 sentences, we only required 150 sentences for re-training. Unfortunately, the re-trained parser still made errors when solving difficult syntactic problems, most notably preposition attachment and preposition ellipsis. PRECISE corrects both types of errors using semantic information.

We refer to PRECISE's use of semantic information to correct parser errors as *semantic over-rides*. Specifically, PRECISE detects that an attachment decision made by the parser is inconsistent with the semantic information in its lexicon.[4] When this occurs, PRECISE attempts to repair the parse tree as follows. Given a noun phrase or a prepositional phrase whose corresponding node n in the parse tree has the wrong parent p, PRECISE traverses the path in the parse tree from p to the root node, searching for a suitable node to attach n to. PRECISE chooses the first ancestor of p such that when n is attached to the new node, the modified parse tree agrees with PRECISE's semantic model. Thus, the semantic over-ride procedure is a generate-and-test search where potential solutions are generated in the order of ancestors of node n in the parse tree. The procedure's running time is linear in the depth of the parse tree.

Consider, for example, the question "What are flights from Boston to Chicago on Monday?" The parser attaches the prepositional phrase "on Monday" to 'Chicago' whereas it should be attached to 'flights' (see Figure 1). The parser merely knows that 'flights', 'Boston', and 'Chicago' are nouns. It then uses statistics to decide that "on Monday" is most likely to attach to 'Chicago'. However, this syntactic decision is inconsistent with the semantic information in PRECISE's lexicon — the preposition 'on' does *not* take a city and a day as arguments, rather it takes a flight and a day.

Thus, PRECISE decides to over-ride the parser and attach 'on' elsewhere. As shown in Figure 1, PRECISE detects that the parser attached the PP "on Monday" to "Chicago" in error. PRECISE attempts to re-attach "on Monday" first to the PP "to Chicago", and then to the NP "flights from Boston to Chicago", where it belongs. While in our example the parser violated a constraint in PRECISE's lexicon, the violation of any semantic constraint will trigger the over-ride procedure.

In the above example, we saw how semantic over-rides help PRECISE fix prepositional attachment errors; they also enable it to correct parser errors in topicalized questions (*e.g.*, "What are Boston to Chicago flights?") and in preposition ellipsis (*e.g.*, when 'on' is omitted in the question "What are flights from Boston to Chicago Monday?").

Unfortunately, semantic over-rides do not correct all of the parser's errors. Most of the remaining parser errors fall into the following categories: relative clause attachment, verb attachment, numeric

[4] We say that node n is *attached* to node p if p is the parent of n in the parse tree.

noun phrases, and topicalized prepositional phrases. In general, semantic over-rides can correct *local* attachment errors, but cannot over-come more global problems in the parse tree. Thus, PRECISE can be forced to give up and ask the user to paraphrase her question.

3 PRECISE Theory

The aim of this section is to explain the theoretical under-pinnings of PRECISE's semantic model. We show that PRECISE always answers questions from the class of Semantically Tractable (ST) questions correctly, given correct lexical and syntactic information.[5]

We begin by introducing some terminology that builds on the definitions given Section 2.

3.1 Definitions

A *join path* is a set of equality constraints between a sequence of database relations. More formally, a join path for relations $R_1, \ldots, R_n$ is a set of constraints $C \subset \{R_i.a = R_{i+1}.b | 1 \leq i \leq n-1\}$. Here the notation $R_i.a$ refers to the value of attribute a in relation R_i.

We say a relation between token set T and a set of database elements and join paths E *respects* a lexicon L if it is a subset of M.

A *question* is simply a string of characters. A *tokenization* of a question (with respect to a lexicon) is an ordered set of strings such that each element of the tokenization is an element of the lexicon's token set, and the concatenation of the elements of the tokenization, in order, is equal to the original question. For a given lexicon and question, there may be zero, one, or several tokenizations. Any question that has at least one tokenization is *tokenizable*.

An *attachment function* is a function $F_{L,q} : T \rightarrow T$, where L is the lexicon, q is a question, and T is the set of tokens in the lexicon. The attachment function is meant to represent dependency information available to PRECISE through a parser. For example, if a question includes the phrase "restaurants in Seattle", the attachment function would attach "Seattle" to "restaurants" for this question. Not all tokens are attached to something in every question, so the attachment function is not a total function. We say that a relation R between tokens in a question q *respects* the attachment function if $\forall t_1, t_2,\ R(t_1, t_2) \Rightarrow (F_{L,q}(t_1) = t_2) \vee (F_{L,q}$ does not take on a value for $t_1)$.

[5]We do not claim that NLI users will restrict their questions to the ST subset of English in practice, but rather that identifying classes of questions as semantically tractable (or not), and experimentally measuring the prevalence of such questions, is a worthwhile avenue for NLI research.

In an NLI, interpretations of a question are SQL statements. We define a *valid interpretation* of a question as being an SQL statement that satisfies a number of conditions connecting it to the tokens in the question. Because of space constraints, we provide only one such constraint as an example: *There exists a tokenization t of the question and a set of database elements E such that there is a one-to-one map from t to E respecting the lexicon, and for each value element $v \in E$, there is exactly one equality constraint in the SQL clause that uses v.*

For a complete definition of a valid interpretation, see (Popescu et al., 2003).

3.2 Semantic Tractability Model

In this section we formally define the class of ST questions, and show that PRECISE can provably map such questions to the corresponding SQL queries. Intuitively, ST questions are "easy to understand" questions where the words or phrases correspond to database elements or constraints on join paths. Examining multiple questions sets and databases, we have found that nouns, adjectives, and adverbs in "easy" questions refer to database relations, attributes, or values.

Moreover, the attributes and values in a question "pair up" naturally to indicate equality constraints in SQL. However, values may be paired with *implicit attributes* that do not appear in the question (*e.g.*, the attribute 'cuisine' in "What are the Chinese restaurants in Seattle?" is implicit). Interestingly, there is no notion of "implicit value" — the question "What are restaurants with cuisine in Seattle?" does not make sense.

A preposition indicates a join between the relations corresponding to the arguments of the preposition. For example, consider the preposition 'from' in the question "what airlines fly from Boston to Chicago?" 'from' connects the value 'Boston' (in the relation 'cities') to the relation 'airlines'. Thus, we know that the corresponding SQL query will join 'airlines' and 'cities'.

We formalize these observations about questions below. We say that a question q is *semantically tractable* using lexicon L and attachment function $F_{L,q}$ if:

1. It is possible to split q up into words and phrases found in L. (More formally, q is tokenizable according to L.)

2. While words may have multiple meanings in the lexicon, it must be possible to find a one-to-one correspondence between tokens in the question and some set of database elements.

(More formally, there exists a tokenization t and a set of database elements and join paths E_t such that there is a bijective function f from t to E_t that respects L.)

3. There is at least one such set E_t that has exactly one wh-value.

4. It is possible to add 'implicit' attributes to E_t to get a set E'_t with exactly one compatible attribute for every value. (More formally, for some E_t with a wh-value there exist attributes $a_1, \ldots, a_n$ such that $E'_t = E_t \cup \{a_1, \ldots, a_n\}$ and there is a bijective function g from the set of value elements (including wh-values) V to the set of attribute elements A in E'_t.)

5. At least one such E'_t obeys the syntactic restrictions of $F_{L,q}$. (More formally, let $A' = A \cap E_t$. Then we require that $\{(f^{-1}(g^{-1}(a)), f^{-1}(a)) \mid a \in A'\}$ respects $F_{L,q}$.)

3.3 Results and Discussion

We say that an NLI is *sound* for a class of questions Q using lexicon L and attachment function F_L if for every input $q \in Q$, every output of the NLI is a valid interpretation. We say the NLI is *complete* if it returns all valid interpretations. Our main result is the following:

Theorem 1 *Given a lexicon L and attachment function F_L,* PRECISE *is sound and complete for the class of semantically tractable questions.*

In practical terms, the theorem states that given correct and complete syntactic and lexical information, PRECISE will return exactly the set of valid interpretations of a question. If PRECISE is missing syntactic or semantic constraints, it can generate extraneous interpretations that it "believes" are valid. Also, if a person uses a term in a manner inconsistent with PRECISE's lexicon, then PRECISE will interpret her question incorrectly. Finally, PRECISE will not answer a question that contains words absent from its lexicon.

The theorem is clearly an idealization, but the experiments reported in Section 4 provide evidence that it is a *useful* idealization. PRECISE, which embodies the model of semantic tractability, achieves very high accuracy because *in practice* it either has correct and complete lexical and syntactic information or it has enough semantic information to compensate for its imperfect inputs. In fact, as we explained in Section 2.1, PRECISE's semantic model enables it to correct parser errors in some cases.

Finding all the valid interpretations for a question is computationally expensive in the worst case (even just tokenizing a question is NP-complete (Popescu et al., 2003)). Moreover, if the various syntactic and semantic constraints are fed to a standard constraint solver, then the problem of finding even a single valid interpretation is exponential in the worst case. However, we have been able to formulate PRECISE's constraint satisfaction problem as a graph matching problem that is solved in polynomial time by the MaxFlow algorithm:

Theorem 2 *For lexicon L,* PRECISE *finds one valid interpretation for a tokenization T of a semantically tractable question in time $\mathcal{O}(Mn^2)$, where n is the number of tokens in T and M is the maximum number of interpretations that a token can have in L.*

4 Experimental Evaluation

Semantic Tractability (ST) theory and PRECISE's architecture raise a four empirical questions that we now address via experiments on the ATIS data set (Price, 1990): how prevalent are ST questions? How effective is PRECISE in mapping ATIS questions to SQL queries? What is the impact of semantic over-rides? What is the impact of parser retraining? Our experiments utilized the 448 context-independent questions in the ATIS "Scoring Set A". We chose the ATIS data set because it is a standard benchmark (see Table 2) where independently generated questions are available to test the efficacy of an NLI.

We found that 95.8% of the ATIS questions were ST questions. We classified each question as ST (or not) by running PRECISE on the question and

System Setup	PRECISE	PRECISE-1
$Parser_{ORIG}$	61.9%	60.3%
$Parser_{ORIG}+$	89.7%	85.5%
$Parser_{TRAINED}$	92.4%	88.2%
$Parser_{TRAINED}+$	94.0%	89.2%
$Parser_{CORRECT}$	95.8%	91.9%

Table 1: **Impact of Parser Enhancements. The** PRECISE **column records the percentage of questions where the small set of SQL queries returned by** PRECISE **contains the correct query;** PRECISE-1 **refers to the questions correctly interpreted if** PRECISE **is forced to return exactly one SQL query. $Parser_{ORIG}$ is the original version of the parser, $Parser_{TRAINED}$ is the version re-trained for the ATIS domain, and $Parser_{CORRECT}$ is the version whose output is corrected manually. System configurations marked by $+$ indicate the automatic use of semantic over-rides to correct parser errors.**

PRECISE	PRECISE-1	AT&T	CMU	MIT	SRI	BBN	UNISYS	MITRE	HEY
94.0%	89.1%	96.2%	96.2%	95.5%	93%	90.6%	76.4%	69.4%	92.5%

Table 2: **Accuracy Comparison between PRECISE , PRECISE-1 and the major ATIS NLIs. Only PRECISE and the HEY NLI are database independent. All results are for performance on the context-independent questions in ATIS.**

recording its response. Intractable questions were due to PRECISE's incomplete semantic information. Consider, for example, the ATIS request "List flights from Oakland to Salt Lake City leaving after midnight Thursday." PRECISE fails to answer this question because it lacks a model of time, and so cannot infer that "after midnight Thursday" means "early Friday morning."

In addition, we found that the prevalence of ST questions in the ATIS data is consistent with our earlier results on the set of 1,800 natural language questions compiled by Ray Mooney in his experiments in three domains (Tang and Mooney, 2001). As reported in (Popescu et al., 2003), we found that approximately 80% of Mooney's questions were ST. PRECISE performance on the ATIS data was also comparable to its performance on the Mooney data sets.

Table 1 quantifies the impact of the parser enhancements discussed in Section 2.1. Since PRECISE can return multiple distinct SQL queries when it judges a question to be ambiguous, we report its results in two columns. The left column (PRECISE) records the percentage of questions where the set of returned SQL queries *contains* the correct query. The right column (PRECISE-1) records the percentage of questions where PRECISE is correct if it is forced to return exactly one query per question. In our experiments, PRECISE returned a single query 92.4% of the time, and returned two queries the rest of the time. Thus, the difference between the two columns is not great.

Initially, plugging the Charniak parser into PRECISE yielded only 61.9% accuracy. Introducing semantic over-rides to correct prepositional attachment and preposition ellipsis errors increased PRECISE's accuracy to 89.7% — the parser's erroneous POS tags still led PRECISE astray in some cases. After re-training the parser on 150 POS-tagged ATIS questions, but without utilizing semantic over-rides, PRECISE achieved 92.4% accuracy. Combining both re-training and semantic over-rides, PRECISE achieved 94.0% accuracy. This accuracy is close to the maximum that PRECISE can achieve, given its incomplete semantic information— we found that, when all parsing errors are corrected by hand, PRECISE's accuracy is 95.8%.

To assess PRECISE's performance, we compared it with previous work. Table 2 shows PRECISE's accuracy compared with the most successful ATIS NLIs (Minker, 1998). We also include, for comparison, the more recent database-independent HEY system (He and Young, 2003). All systems were compared on the ATIS scoring set 'A', but we did "clean" the questions by introducing sentence breaks, removing verbal errors, *etc.*. Since we could add modules to PRECISE to automatically handle these various cases, we don't view this as significant.

Given the database-specific nature of most previous ATIS systems, it is remarkable that PRECISE is able to achieve comparable accuracy. PRECISE does return two interpretations a small percentage of the time. However, even when restricted to returning a single interpretation, PRECISE-1 still achieved an impressive 89.1% accuracy (Table 1).

5 Related Work

We discuss related work in three categories: Database-independent NLIs, ATIS-specific NLIs, and sublanguages.

Database-independent NLIs There has been extensive previous work on NLIs (Androutsopoulos et al., 1995), but three key elements distinguish PRECISE. First, we introduce a model of ST questions and show that it produces provably correct interpretations of questions (subject to the assumptions of the model). We measure the prevalence of ST questions to demonstrate the practical import of our model. Second, we are the first to use a statistical parser as a "plug in", experimentally measure its efficacy, and analyze the attendant challenges. Finally, we show how to leverage our semantic model to correct parser errors in difficult syntactic cases (*e.g.*, prepositional attachment). A more detailed comparison of PRECISE with a wide range of NLI systems appears in (Popescu et al., 2003). The advances in this paper over our previous one include: reformulation of ST THEORY, the parser retraining, semantic over-rides, and the experiments testing PRECISE on the ATIS data.

ATIS NLIs The typical ATIS NLIs used either domain-specific semantic grammars (Seneff, 1992; Ward and Issar, 1996) or stochastic models that required fully annotated domain-specific corpora for reliable parameter estimation (Levin and Pieraccini, 1995). In contrast, since it uses its model of semantically tractable questions, PRECISE does not

require heavy manual processing and only a small number of annotated questions. In addition, PRECISE leverages existing domain-independent parsing technology and offers theoretical guarantees absent from other work. Improved versions of ATIS systems such as Gemini (Moore et al., 1995) increased their coverage by allowing an approximate question interpretation to be computed from the meanings of some question fragments. Since PRECISE focuses on high precision rather than recall, we analyze every word in the question and interpret the question as a whole. Most recently, (He and Young, 2003) introduced the HEY system, which learns a semantic parser without requiring fully-annotated corpora. HEY uses a hierarchical semantic parser that is trained on a set of questions together with their corresponding SQL queries. HEY is similar to (Tang and Mooney, 2001). Both learning systems require a large set of questions labeled by their SQL queries—an expensive input that PRECISE does not require—and, unlike PRECISE, both systems cannot leverage continuing improvements to statistical parsers.

Sublanguages The early work with the most similarities to PRECISE was done in the field of sublanguages. Traditional sublanguage work (Kittredge, 1982) has looked at defining sublanguages for various domains, while more recent work (Grishman, 2001; Sekine, 1994) suggests using AI techniques to learn aspects of sublanguages automatically. Our work can be viewed as a generalization of traditional sublanguage research. We restrict ourselves to the semantically tractable subset of English rather than to a particular knowledge domain. Finally, in addition to offering formal guarantees, we assess the prevalence of our "sublanguage" in the ATIS data.

6 Conclusion

This paper is the first to provide evidence that statistical parsers can support NLIs such as PRECISE. We identified the quandary associated with appropriately training a statistical parser: without special training for each database, the parser makes numerous errors, but creating a massive, labeled corpus of questions for each database is prohibitively expensive. We solved this quandary via light re-training of the parser's tagger and via PRECISE's semantic over-rides, and showed that in concert these methods enable PRECISE to rise from 61.9% accuracy to 94% accuracy on the ATIS data set. Even though PRECISE is database independent, its accuracy is comparable to the best of the database-specific ATIS NLIs developed in previous work (Table 2).

References

I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. 1995. Natural Language Interfaces to Databases - An Introduction. In *Natural Language Engineering, vol 1, part 1*, pages 29–81.

E. Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proc. of NAACL-2000*.

R. Grishman. 2001. Adaptive information extraction and sublanguage analysis. In *Proc. of IJCAI 2001*.

B.J. Grosz, D. Appelt, P. Martin, and F. Pereira. 1987. TEAM: An Experiment in the Design of Transportable Natural Language Interfaces. In *Artificial Intelligence 32*, pages 173–243.

Y. He and S. Young. 2003. A data-driven spoken language understanding system. In *IEEE Workshop on Automatic Speech Recognition and Understanding*.

R. Kittredge. 1982. Variation and homogeneity of sublanguages. In R. Kittredge and J. Lehrberger, editors, *Sublanguage: Studies of Language in Restricted Semantic Domains*, pages 107–137. de Gruyter, Berlin.

E. Levin and R. Pieraccini. 1995. Chronus, the next generation. In *Proc. of the DARPA Speech and Natural Language Workshop*, pages 269–271.

W. Minker. 1998. Evaluation methodologies for interactive speech systems. In *First International Conference on Language Resources and Evaluation*, pages 801–805.

R. Moore, D. Appelt, J. Dowding, J. M. Gawron, and D. Moran. 1995. Combining linguistic and statistical knowledge sources in natural-language processing for atis. In *Proc. of the ARPA Spoken Language Technology Workshop*.

A. Popescu, O. Etzioni, and H. Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proc. of IUI-2003*.

P. Price. 1990. Evaluation of spoken language systems: the atis domain. In *Proc. of the DARPA Speech and Natural Language Workshop*, pages 91–95.

S. Sekine. 1994. A New Direction For Sublanguage NLP. In *Proc. of the International Conference on New Methods in Language Processing*, pages 165–177.

S. Seneff. 1992. Robust parsing for spoken language systems. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*.

L.R. Tang and R.J. Mooney. 2001. Using Multiple Clause Constructors in Inductive Logic Programming for Semantic Parsing. In *Proc. of the 12th European Conference on Machine Learning (ECML-2001), Freiburg, Germany*, pages 466–477.

W. Ward and S. Issar. 1996. Recent improvements in the cmu spoken language understanding system. In *Proc. of the ARPA Human Language Technology Workshop*, pages 213–216.