Extrinsic Parse Selection

David Goss-Grubbs
University of Washington
Department of Linguistics
Box 354340
Seattle, WA 98195-4340, USA
davidgg@u.washington.edu

Abstract

This paper reports on one aspect of Locutus, a natural language interface to databases (NLIDB) which uses the output of a high-precision broad-coverage grammar to build semantic representations and ultimately SQL queries. Rather than selecting just a subset of the parses provided by the grammar to use in further processing, Locutus uses all of them. If the meaning of a parse does not conform to the semantic domain of the database, no query is built for it. Thus, intended parses are chosen extrinsically. The parser gives an average of 3.01 parses to the sentences in the GEOQUERY250 corpus. Locutus generates an average of 1.02 queries per sentence for this corpus, all of them correct.

1 Introduction

Natural language sentences are typically more ambiguous than the people who utter them or perceive them are aware of. People are very good at using context and world knowledge to unconsciously disambiguate them. High-precision, broad-coverage grammars, however, often assign every legitimate analysis to a given sentence, even when only one of them reflects the sentence's intended meaning. It is thus important for natural language processing applications that use these analyses to be able to reliably select the intended parse. It is typical for such applications to choose the best parse up front and pass just that one on to further processing. For some applications, however, it is possible, and indeed preferable, to pass all the parses on and let downstream processing decide which parses to use.

This paper describes such an application. Locutus (Goss-Grubbs to appear), a natural language interface to relational databases (NLIDB), creates semantic representations for the parses assigned by a high-precision broad-coverage grammar, and from those creates SQL queries. It does not include a step where one or more "best" parses are selected for further processing. Queries are built for all parses for which it is possible to do so. For a standard corpus of NLIDB training sentences, it is able to generate the correct query whenever a suitable analysis is given by the parser. In the rare case where it generates two queries, both queries are equally correct.

2 Parse Selection

Parse selection for probabilistic grammars involves simply finding the most probable parse, or top-N most probable parses, and can be done using efficient algorithms, (e.g. Klein and Manning, 2003).

Things are different for high-precision, hand-coded grammars, such as the LinGO English Resource Grammar, ERG (Flickinger, 2000), a Head-Driven Phrase Structure Grammar implementation of English; and Xerox's English grammar (Butt, et al., 2002), a Lexical Functional Grammar implementation. These grammars do not define a probability distribution over parses. Rather, they assign to each string all of its grammatically valid

Proceedings of the NAACL HLT 2010 Student Research Workshop, pages 19–22,
Los Angeles, California, June 2010.

parses. Techniques for deciding between parses produced by these kinds of grammars include using sortal constraints on arguments of semantic relations (Müller and Kasper, 2000); and annotating individual grammatical rules with weights (Kiefer, et al., 1999). More recently, the development of rich treebanks such as the LinGO Redwoods (Oepen, et al., 2004) which stores all analyses of a sentence, along with an indication of which is the preferred one, makes it possible to train maximum entropy models for parse selection, (e.g. Toutanova, et al., 2002).

For at least the NLIDB task, however, selection of the best parse is not an end in itself. Rather, what is necessary is to generate the intended database query. Indeed, two or more distinct syntactic parses may all lead to the same (intended) query. If the NLIDB identifies this query correctly, it has achieved its goal without, strictly speaking, having selected the best parse.

Furthermore, eliminating any grammatically valid parse without subjecting it to further processing risks missing the intended query. For these reasons, Locutus does no intrinsic parse selection. Rather, it tries to build a query for all valid parses. The semantic constraints of the database domain limit well-formed semantic representations to those that make sense in that domain, so that a grammatically valid parse may not receive a legitimate semantic representation, and thus not receive a database query.

3 Locutus

Locutus is an NLIDB which is designed to be portable with respect to source language and grammatical formalism. It can take as input the syntactic analyses produced by any sufficiently sophisticated grammar/parser. The implementation reported on in this paper consumes the f-structures produced by the Xerox English grammar.

Locutus is also portable with respect to database domain. The projection of semantic structures from the syntactic analyses provided by the parser is guided by a semantic description of the database domain together with a set of constraints called *sign templates* linking syntactic patterns with semantic patterns.

High precision (building only correct queries) is maintained in a number of ways:

- High-precision syntactic grammars are used.
- The projection of semantic structures from syntactic structures is resource-sensitive. Every element of the syntactic structure must be referenced just once by the sign template that licenses the corresponding semantic structure.
- The semantic description of the database domain defines a network of semantic relationships and their arguments, along with constraints regarding which arguments are compatible with one another. In this way, semantic structures which would otherwise be generated can be ruled out.

3.1 Processing Pipeline

The processing of a sentence by Locutus proceeds in the following way. The string of words is passed to the XLE parser, which returns a contextualized feature structure from which individual parses are extracted. An example parse appears in Figure 1.

```
[ PRED border
  SUBJ [ PRED state
         NTYPE [ NSYN common ]
         SPEC [ DET [ PRED which
                      NTYPE [ NSYN … ]
                      PRON-TYPE int ] ]
         CASE nom
         NUM pl
         PERS 3 ]
  OBJ [ PRED delaware
        NTYPE [ NSYN proper ]
        CASE obl
        NUM sg
        PERS 3 ]
  PRON-INT [...]
  FOCUS-INT [...]
  TNS-ASP [...]
  CLAUSE-TYPE int
  PASSIVE -
  VTYPE main ]
```

Figure 1: parse for “Which states border delaware?”

Locutus interprets this syntactic analysis into a set of semantic representations called Semantic Mobile Structures, an example of which appears in an abbreviated form in Figure 2.

```
x0 DefQuant: [ > [1]]
   r0 Border:STATE1
        STATE2: x1 DefQuant: [1]
                   r1 StateName:STATE
                        NAME: [delaware]
   r2 State:STATE
```

Figure 2: SMS for "Which states border delaware?"

Finally, this representation is translated into an SQL query, as shown in Figure 3, which is sent to the database, and the answer is shown.

```
select t1.Name
from border, state t1, state t2
where border.State1 = t1.Name and
      border.State2 = t2.Name and
      t2.Name = 'delaware'
```

Figure 3: query for "Which states border Delaware?"

3.2 Efficiency

There is a bit of time savings in not having an intrinsic parse-selection step. These savings are counterbalanced by the extra time it takes to interpret parses that would have otherwise been excluded by such a step. However, a certain amount of syntactic structure is shared among the various parses of a syntactically ambiguous sentence. Locutus recognizes when a piece of syntactic structure has already been interpreted, and reuses that interpretation in every parse in which it appears. In this way Locutus minimizes the extra time taken to process multiple parses. At any rate, processing speed does not appear to be a problem at this point in the development of Locutus.

3.3 Further Work

Although Locutus has a wide range of functionality, it is still a work in progress. The format for authoring sign templates is rather complex, and customizing Locutus for a given database can be time-consuming. I anticipate an authoring tool which makes much of the customization process automatic, and hides much of the complexity of the rest of the process from the author, but such a tool has yet to be implemented.

4 Experiment

To test the coverage and precision of Locutus, I have customized it to answer questions from the GEOQUERY 250 corpus (Mooney, 1996), which consists of a database of geographical information paired with 250 English sentences requesting information from that database. 25 of these sentences are held out for the purposes of another study, and I have not examined the behavior of Locutus with respect to these sentences. I ran the other 225 sentences through Locutus, keeping track of which sentences Locutus built at least one query for. For each of those sentences, I also tracked the following:

- How many syntactic parses were generated by the grammar
- How many queries were produced
- How many of those queries were correct

The XLE Engine includes a facility to do stochastic disambiguation (Kaplan, et al. 2004), and the English grammar I used comes with a property weights file of the kind required by the disambiguation process. I ran the sentences through Locutus using just the single best parse returned by that process, keeping track of how many queries were produced.

5 Results

223 of the 225 sentences (99.1%) are assigned at least one query. For the other two sentences, no analysis returned by the parser reflect the intended meaning of the sentence. The average number of parses for these sentences is 3.01, with 158 sentences given at least two parses, and 84 sentences given at least three. Some sentences were given as many as 20 parses.

Figure 4 contains the graph of the number of parses by the average number of queries assigned to sentences with that many parses. Note that the number of queries per sentence is not correlated with the number of parses assigned by the grammar. The sentences that were assigned more than one query were each assigned either one or two parses. All the sentences with more syntactic parses were assigned a single query each.

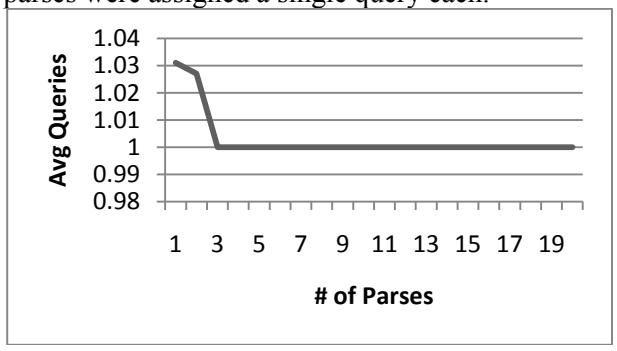

Figure 4: Average queries by ambiguity level

Of the 223 sentences that were assigned a query, 219 of them were assigned exactly one query. Every query was correct in the sense that it accu-

rately reflected a reasonable interpretation of the sentence. Four sentences were each assigned two queries. They are given in (1)-(4).

(1) How many people live in Washington?
(2) How many people live in New York?
(3) What is the length of the Colorado river?
(4) What is the length of the Mississippi river?

It is appropriate that each of these sentences gets two queries. For (1)-(2), the GEOQUERY 250 database contains cities, their populations, states and their populations; "Washington" and "New York" are both names of cities and states that appear in the database. For (3)-(4), one interpretation is to return the length of the river mentioned in the sentence. The other possibility is to return all the rivers that are the same lengths as the ones mentioned. For instance, in the GEOQUERY database, the Colorado and Arkansas rivers are both 2333 km long. One valid answer to (3) is the number "2333". The other valid answer is the list of rivers "Arkansas" and "Colorado". To give any of these sentences only a single query would be to miss a reasonable interpretation.

Table 1 summarizes the results when only a single parse for each sentence, chosen stochastically using the property weights file provided with the XLE English grammar, is sent to Locutus. The parse is considered correct if it leads to a correct query.

	# of sents	avg. parses	% correct
≥ 1 parse	223	3.01	54%
≥ 2 parses	158	3.84	35%

Table 1

Although performance is better than chance, it is clearly less successful than when Locutus is allowed to use every parse, in which case a correct query is always constructed.

6 Conclusion

For natural language processing applications that take the results of a high-precision syntactic parser and pass them along to further processing, selecting the correct parse is not an end in itself. It is only useful insofar as it improves the final result.

For applications such as NLIDBs, which are provided with a precise semantic framework within which sentences may be interpreted, it is better to pass along the full set of grammatically valid parses than to select beforehand a limited subset of those parses. Using this technique, Locutus achieves 100% correctness on the sentences for which it builds a query.

References

Butt, Miriam, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. "The Parallel Grammar Project." *Proceedings of COLING2002 Workshop on Grammar Engineering and Evaluation.* 2002.

Flickinger, Dan. "On building a more efficient grammar by exploiting types." *Natural Language Engineering* 6, no. 1 (2000): 15-28.

Goss-Grubbs, David. "Deep Processing for a Portable Natural Language Interface to Databases." *dissertation, University of Washington.* to appear.

Kaplan, Ron, Stefan Riezler, Trace King, John Maxwell, Alexander Vasserman, and Richard Crouch. "Speed and Accuracy in Shallow and Deep Stochastic Parsing." *Proceedings of the Human Language Technology Conference and the 4th Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'04).* Boston, MA, 2004.

Kiefer, Bernd, Hans-Ulrich Krieger, John Carroll, and Rob Malouf. "A Bag of Useful Techniques for Efficient and Robust Parsing." *Proceedings of the 37th Meeting of the Association for Computational Linguistics.* College Park, MD, 1999. 473-480.

Klein, Dan, and Christopher D. Manning. "A* Parsing: Fast Exact Viterbi Parse Selection." *Proceedings of HLT-NAACL 2003.* 2003. 40-47.

Mooney, Raymond. *Geoquery Data.* 1996. http://www.cs.utexas.edu/users/ml/nldata/geoquery.html (accessed February 13, 2010).

Müller, Stefan, and Walter Kasper. "HPSG Analysis of German." In *Verbmobil. Foundations of Speech-to-Speech Translation*, edited by Wolfgang Wahlster, 238-253. Berlin: Springer, 2000.

Oepen, Stephan, Dan Flickinger, Kristina Toutanova, and Christopher D. Manning. "LinGO Redwoods: A Rich and Dynamic Treebank for HPSG." *Research on Language and Computation* (Springer) 2 (2004): 575-596.

Toutanova, Kristina, Christopher D. Manning, Stuart Shieber, Dan Flickinger, and Stephan Oepen. "Parse disambiguation for a rich HPSG grammar." *Proceedings of the First Workshop on Treebanks and Linguistic Theories.* 2002. 253-263.