

A Phrasal Approach to Natural Language Interfaces over Databases

Michael Minock

Department of Computing Science
Umeå University, Sweden 90187
Phone: +46 90 786 6398 FAX: +46 90 786 6126
Email: mjm@cs.umu.se

Abstract. This short paper introduces the STEP system for natural language access to relational databases. In contrast to most work in the area, STEP adopts a phrasal approach; an administrator couples phrasal patterns to elementary expressions of tuple relational calculus. This ‘phrasal lexicon’ is used bi-directionally, enabling the generation of natural language from tuple relational calculus and the inverse parsing of natural language to tuple calculus. This ability to both understand and generate natural language enables STEP to engage the user in clarification dialogs when the parse of their query is of questionable quality or is open to multiple interpretations. An on-line demonstration of STEP is accessible at <http://www.cs.umu.se/~mjm/step>.

1 Introduction

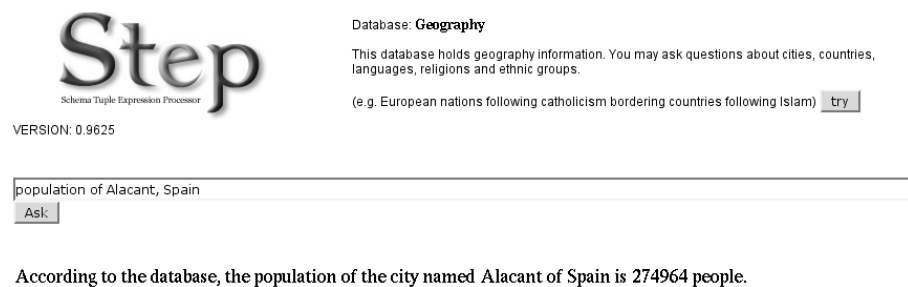
To get a grip on the problem of reliable natural language interfaces, we focus on the case in which the information of user interest is housed within relational databases and interaction is purely textual; user requests are single sentences of natural language and answers are multiple sentences of natural language. Clarification dialogs, when necessary, are limited to multiple choice, yes/no responses from the user. Historically projects with such assumptions have aroused great interest within the relational database and computational linguistics communities [1, 4] which has continued, albeit with less fervor, into the current period (see for example [13, 2, 14]). RENDEZVOUS [3] was perhaps the first project proposed¹ and, to the point here, Codd was very explicit about requiring his system to engage the user in clarification dialogs; single shot systems where requests are parsed to a formal language and then immediately applied to a back-end database were deemed likely to be misinterpreted and misprized by users. At the very least a system should be able to paraphrase the user’s query back to them during answer presentation or ambiguity resolution. In short, *any practical natural language interface over databases must have query paraphrasing capabilities.*

¹ It is doubtful whether RENDEZVOUS was completed given the limited facilities of the time.

It would be wrong to claim that other projects did not address the paraphrasing problem, some did [7, 6]. But as approaches switched from semantic grammars to those using domain independent grammars [5], the focus on generating query paraphrases tended to be discarded. Though the idea of doing a full, bi-directional integration of domain independent grammars is appealing, one has to wonder if this is currently feasible for interfaces to databases; the problems of ambiguity in large scale grammars, of configuring mappings between domain independent logical form and database relations, of capturing idiosyncratic domain language and of making the whole system bi-directional give one pause.

In response to this, STEP adopts a phrasal approach to the configuration and maintenance of linguistic knowledge. Specifically an administrator authors a *phrasal lexicon* by coupling phrasal patterns to elementary expressions of a class of tuple calculus. This phrasal lexicon is used bi-directionally, enabling the generation of natural language from tuple relational calculus and the inverse parsing of natural language to tuple calculus. This ability to both understand and generate natural language enables STEP to engage the user in clarification dialogs when the parse of their query is of questionable quality or is ambiguous. The details of STEP are documented in a recent technical report [12] available at <http://www.cs.umu.se/~mjm/step>.

2 The Web Interface



The screenshot shows the STEP web interface. On the left, the logo "Step" is displayed in a large, stylized font, with "Schema Tuple Expression Processor" written below it. Underneath the logo, the text "VERSION: 0.9625" is visible. To the right of the logo, the database name "Database: Geography" is shown. Below this, a descriptive paragraph states: "This database holds geography information. You may ask questions about cities, countries, languages, religions and ethnic groups." A small example query "(e.g. European nations following catholicism bordering countries following Islam)" is followed by a "try" button. Below the description, there is a search input field containing the text "population of Alacant, Spain" and an "Ask" button. At the bottom of the interface, the result of the query is displayed: "According to the database, the population of the city named Alacant of Spain is 274964 people."

Fig. 1. Basic Querying.

STEP is currently about 10,000 lines of LISP code, run as a server and accessed via standard Internet browsers. STEP issues satisfiability queries to the SPASS theorem prover and relational queries to a PostgreSQL database. Recently WordNet [9] has also been integrated into the system. STEP supports concurrent users and has responses times in the neighborhood of 2 to 5 seconds for queries over a geography database [8]. Figure 1 shows the web-based interface. Note that the current database is the **Geography** database, which has a simple canned paragraph description. Users enter their requests on the input field and

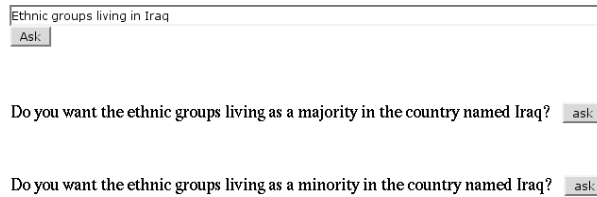


Fig. 2. A Clarification Dialog.

obtain answers in the area immediately below. Figure 2 shows the response to a user request that is open to multiple interpretations.

A full demonstration of STEP over a geography database, complete with a link to its complete configuration, has been continuously available for anonymous querying since June 2004. Over this period approximately 150 different visitors (myself not included) have issued in the neighborhood of 1000 queries to STEP. The purpose of this initial testing was not to carefully measure the accuracy of STEP, but was rather to build up a relatively large sample of real requests and in doing so help determine where system development efforts should be placed; naturally STEP has undergone extensive development and refinement over this period. Still, based on a cursory analysis of more recent system logs, STEP has done a reasonably accurate job of satisfying user requests. In the future more systematic studies will be undertaken over different domains to quantify this.

3 Discussion

The architectural of STEP is somewhat unusual. STEP does not use a domain independent grammar for syntactic analysis, but rather a phrasal lexicon authored specifically over the underlying database. Input sentences are parsed via a closed set of inference rules [12]. These inference rules employ a Montague like strategy where ‘logical form’ is incrementally built up as the input is scanned from left to right. These inference rules also cause special *fudging operations* which, at a cost, deliberately add, drop or alter words in the input sentence to help find a parse, albeit one of less confidence. To prepare for query paraphrasing, STEP compiles the phrasal lexicon into a subsumption hierarchy. Paraphrasing a given logical query comes down to semantically sorting the query into this hierarchy and gathering phrasal attachments of the sorted query’s immediate parents [11].

‘Logical form’ in STEP consists of expressions in a class of tuple calculus with attached pragmatic features. The actual class of tuple calculus is decidable for emptiness, containment and equivalence [10]. Such capabilities are used in paraphrasing and in reasoning to support cooperative responses [11]. The relations used within these expressions are over database as well as pragmatic and conceptual relations. Such an extended vocabulary of relations makes it possible that STEP might provide meta level responses for queries over meaningful entities and relationships not (yet) covered in the underlying database.

The advantage of STEP's phrasal approach is that it avoids many of the difficulties associated with ambiguity in large scale domain independent grammars and maps directly to the underlying database relations. Additionally, via fudging operations, STEP finds acceptable parses for many non-grammatical inputs, a common occurrence in practice. Finally a phrasal approach allows for easier specification of idiomatic and idiosyncratic domain language. A disadvantage of STEP's approach is that a specific phrasal lexicon must be authored for each new database. That said, we envision tools to assist in this process and we note that entries in the phrasal lexicon are relatively well structured and can be compiled into a subsumption hierarchy which makes their semantic relationships explicit. A question of course, is how well will STEP handle the syntactic complexities of real language over more complex databases. Of course it is still too early to answer this question fully, but the thesis here is that through better integrating large electronic dictionaries into the parsing process and through relying on clarification dialogs, STEP will ultimately work well enough to be practical.

References

1. I. Androutsopoulos and G.D. Ritchie. Database interfaces. In R. Dale, H. Moisl, and H. Somers, editors, *Handbook of Natural Language Processing*, pages 209–240. Marcel Dekker Inc., 2000.
2. A. Blum. Microsoft english query 7.5: Automatic extraction of semantics from relational databases and OLAP cubes. In *Proc. of VLDB*, pages 247–248, 1999.
3. E. Codd. Seven steps to rendezvous with the casual user. In *IFIP Working Conference Data Base Management*, pages 179–200, 1974.
4. A. Copestake and K. Sparck Jones. Natural language interfaces to databases. *The Natural Language Review*, 5(4):225–249, 1990.
5. B. Grosz, D. Appelt, P. Martin, and F. Pereira. Team: An experiment in the design of transportable natural-language interfaces. *AI*, 32(2):173–243, 1987.
6. J. Ljungberg. Paraphrasing SQL to natural language. In *Proc. of RIAO 91*, Barcelona, 1991.
7. B. Lowden and A. de Roeck. The REMIT system for paraphrasing relational query expressions into natural language. In *Proc. of VLDB*, pages 365–371, 1986.
8. W. May. Information extraction and integration with FLORID: The MONDIAL case study. Technical Report 131, Universität Freiburg, Institut für Informatik, 1999.
9. G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Five papers on WordNet. Technical report, Princeton University, Princeton, N.J, 1993.
10. M. Minock. Knowledge representation using schema tuple queries. In *Proc. of KRDB*, pages 51–62, Hamburg, Germany, 2003. IEEE Computer Society Press.
11. M. Minock. Modular generation of relational query paraphrases. *Journal of Language and Computation special issue on Formal Aspects of NLG*, 2005. To appear.
12. M. Minock. A phrasal approach to natural language access over relational databases. Technical Report 05.09, Umeå University, Umeå, Sweden, March 2005.
13. A. Popescu, O. Etzioni, and H. Kautz. Towards a theory of natural language interfaces to databases. In *Intelligent User Interfaces*, 2003.
14. B. Thalheim and T. Kobienia. Generating DB queries for web NL requests using schema information and DB content. In *Proc. of NLDB*, pages 205–209, 2001.