A Natural Language Query Processor for a Database

Milo Velimirović
Kasi Periyasamy
Computer Science Department
University of Wisconsin – La Crosse
La Crosse WI 54601
milov@cs.uwlax.edu

Abstract

This paper describes a recognizer based on a subset of English that is simple but limited to the chosen problem domain. The lexical inventory and English grammar were built up incrementally using LEX and YACC compiler tools. The recognizer extracts the necessary information to fill-in clauses of an SQL template that when completed and run as a query against a database system answers the question originally stated in English.

The benefit of a natural language query processor is that no knowledge of database structure is required to use it. A user needs only to know what is in the database, but not how it is organized. In the current work variations of 'who' questions are recognized in the context of a small movie database. The recognizer could be easily extended by the addition of lexical and grammar rules and SQL templates to address different questions, such as 'how many'.

1. Introduction

Since their inception digital computing devices have been used in natural language processing. Among the first electronic digital computers was the Colossus (Flowers 1983), which was used by the British to decipher encrypted enemy communications during World War II. The Colossus had no understanding of the messages it was processing. It merely performed Boolean operations on data from paper tapes. In the 1950s a project at Harvard sought to use computers to perform automatic translation of text from Russian to English (Oettinger 1960). Unfortunately, the automatic translation of texts never became a successful endeavor.

Numerous languages have been developed to facilitate human-computer interactions to allow humans to more effectively and efficiently direct computers to solve problems that they were interested in. The research in language theory led to the creation of tools that permit the rapid construction of parsers, software that can recognize correctly formed statements of a specified computer language. Perhaps the most well-known of these tools is Yacc: Yet Another Compiler-Compiler, (Johnson 1978), which is available on most computers running UNIX.

While the tools for computer languages were being developed, the storage capacity of computers was being increased as well. With the increase in data storage capacity, new techniques for managing data entry, manipulation and retrieval were invented. The most successful of these techniques are database management systems that implement E. F. Codd's relational database model (1970) in which data are stored as unique rows organized into tables. Most relational database systems provide access through Structured Query Language or SQL. The advantage of the relational model and

SQL is the great increase in productivity it affords. Unfortunately, SQL is a *computer* programming language—its syntax and intricacies must be learned if one wished to use it to make queries of a database. Despite all the advances in technology, there remains a gap between the well understood problems associated with designing a synthetic, computer programming language and writing software that will parse and understand more than a small subset of a natural language. The difference can be illustrated by a comparison of SQL to English:

SQL statement:

```
select f.director from films f where f.title = "Edward
Scissorhands";
```

English statement:

"Who directed the movie, 'Edward Scissorhands'?"

In either method of asking the question, the desired result or answer is 'Tim Burton'.

This paper describes a project that explored the use of parser generation tools that facilitate the mapping and translation of natural language questions in a subset of English language into database queries using Structured Query Language. The mapping between simple questions such as, "Who directed… ?" or "When was… ?" can be made in a straightforward manner. This would require a style of inquiry where all of the questions asked are self-contained. Most conversations contain references to things that have been mentioned at some other time or to persons by using pronouns. These serve as a shorthand, or placeholder for something or someone previously mentioned, but add a component of natural language to a query. For example, it would be desirable to be able to follow the above example query with, "When was it released?" and have "it" or the phrase, "the movie," correctly resolve to the movie title.

. In the current project, the types of English questions will begin with simple "newspaper" questions of who, what, where, when and how (many) —why will be left as an exercise for the reader. Since a conversational style is a feature of natural languages, questions that employ compound conditions, e.g. "What movies were written, produced and directed by Sam Fuller?" will be investigated as well as those that employ pronouns.

2. Literature Review

The desire to interact in a straightforward, natural way with computers to answer questions about data is as old as the computer technology itself. The earliest computer programming languages were an attempt to give computer users a notation for stating the methods to be used to solve their problems that was more familiar than the codes for the machine instructions performed by the computers. These languages still required a significant investment in time and training for one to become proficient in their use. The vocabulary of these languages was limited and the syntax of statements was inflexible; in large part, these restrictions existed to simplify the process of translating programming languages into computer machine instructions.

The first programming languages were translated into machine instructions in an ad hoc fashion, without a formal theory to guide the process. Computer scientists developed the theories, techniques and tools needed to completely describe and specify programming languages. These included automata, formal grammars, parsers and analyzers. These allowed construction of *scanner* software to recognize the numbers, words and punctuation; *parsers* based on grammars permitted software to analyze the statements of a programming language so that they could be further processed and

translated into the machine instructions that that could themselves be run on computers. Natural languages, such as English, have a large vocabulary, and do not have a syntax that can be reduced to a number of grammar rules in a manner that directly parallels that of programming languages. This makes the task of translating English for purposes of controlling or querying computers much more difficult than using programming languages. One way to simplify the problem is to restrict the types and formats of the language that can be used in communications with a computer. Another way to simplify the problem is often enforced by the domain of inquiry, shrinking the vocabulary to cover only the facts or data contained in a database. It's almost always easier to solve a problem that is narrowly defined, than to attempt to construct a universal problem solver.

The research reported in this paper begins with investigations that use simple English constructions to engage in conversations with software about very restricted domains of knowledge. Over time the types of queries increase in complexity, but the domains remain quite limited. Once the relational database model took hold as the predominant methodology for data representation and the use of Structured Query Language (SQL) became the standard method of interacting with relational databases, research focused on converting natural-languages inquiries into SQL.

In a 1968 revised version of his dissertation, Bobrow discusses the reasons for wanting to use natural languages rather than computer programming languages to interact with a computer. His reasons from the 1960s are still valid. The reasons include avoiding the need to train a casual user of computers in a programming language and "Programming languages.... cannot describe a problem, only a method for finding a solution to the problem" (p 148-9).

He continues by setting up a framework and "criteria for evaluating question answering systems" (p 150). The four criteria measure the level or depth of understanding of a natural language, the extensibility of the system, the level of knowledge of the internal structure of the system by its users and users' interactions with the system.

Bobrow proceeds into a detailed analysis of communication, both spoken and written. He introduces the concept of kernel sentences, those simplest sentences, "the listener can understand directly" (p158). He considers both the generation of sentences and the analysis (parsing) of sentences at multiple levels: syntactic, semantic and deductive. The deductive component is Bobrow's software, STUDENT, in its use of facts to return answers to a user's question(s).

Hendrix, et al., describe a query system, whose domain of knowledge is Navy data. In their paper they give examples of numerous simple queries, "How long is the Philadelphia[?]" (1978, p 110). This would appear to be what Bobrow describes as a kernel sentence. It consists of a query of a single fact about a single item in the database. They include a section on special features of their software that include a spelling corrector, e.g. "Constallation" is corrected to "Constellation" (p 112,124), the ability to process elliptical queries, those that refer to a portion of a prior query.

Sidner (1979) shows how to parse sentences that contain anaphoric references in the context of a conversation. One common situation is resolving the referent of pronouns. Her research looks at sentences as part of a conversation which provides context and not just as isolated statements.

Up to this time, researchers presented examples using programs in the programming langauage LISP, with little mention of the mechanisms used to store data.

Petrick (1982) demonstrates the parsing of English queries with LISP, but creates SQL queries to be run against a relational database. The queries are no longer just simple fact based queries, but complex so-called aggregate functions in databases, those that require some operation be performed on selected elements of the database. Additionally Petrick presents methods for parsing questions with compound conditions.

Thompson et al. 1983 describe a menu-based system that allows end users to construct queries in English. The guided nature of the menu-based approach assures that the question will be well formed and that the application will be able to construct a database query. As the technology of ordinary interaction changes, so does the methodology for creating queries.

Tomita 1987 details the use of a context free grammar (CFG) to parse English questions. He explains the extensions that are needed to be able to parse natural languages and also the data structures that result from his approach. His algorithm for parsing uses a complex data structure that he calls a "graph-structured stack," (p31) where a traditional parser would use a simple stack of elements with no additional structure.

Johnson and Bryant take relaxed approach to parsing English for queries, recognizing that even if a query is ungrammatical English it may still contain all the necessary components to construct a query that will successfully run against a database. They give the example "Who die 1787?" and describe it as "ungrammatical, but meaningful as a request for information" (Johnson & Bryant, 1994, p 222).

Hallett (2006) describes a portable system for generating natural language queries against a database. The novel feature of this system is that it will pre-compute the

possible options available to a user of the system and present them in a menu or table driven user interface. This would appear to be a modern implementation of Thompson's (1983) approach.

In a recently published interview, Marvin Elder of Semantra describes how his company's products using "Conversational Analytics" exceeds the convention business intelligence tools currently available by being able to "redefine concepts with business jargon and abbreviations … [to] get real-world conversation between non-technical users and the enterprise data." (Erickson, 2008-March, np). This illustrates the path of many technologies that begin as research on basic questions and end up as a product.

On research methodology, Vassilou et al. present a study of computer users creating database queries using a natural language tool and creating queries directly using SQL. They present a detailed approach to both forming hypotheses and the analyses that should be conducted on the differences between natural language and SQL queries.

The literature shows that research tracks both the communities where it is conducted and uses the technologies available at the time. LISP was the lingua franca of the Artificial Intelligence community at MIT where both Bobrow and Sidner worked and thus it is no surprise to find all of their examples in LISP. Once database technology had been standardized using the relational model and SQL, accesses to databases was almost always through the intermediate step of using SQL. Additional literature needs to be reviewed to determine what additional standard tools of computer science can be brought to bear on the problem of parsing English, or any natural language, for the purpose of constructing database queries.

3. Methodology

The English language allows one to ask a question in many different ways. All of the various forms of the question may have the same answer. While it may be easy for a speaker of the language to recognize the equivalence of the questions, it is not necessarily so for a computer program that has the task of recognizing questions. In contrast there may be only a few or even only one (preferred) way to formulate a relational database query. In order to investigate the tools and methods needed to translate English questions into queries that can be run against a relational database management system, it is necessary to have a number of questions that can be used as test cases. The initial problem investigated was simple "newspaper" questions of who, what, where, when and how (many) and what is necessary in the grammar rules to be able to successfully parse these questions.

In this study a relational database with a small number of tables was used to explore these questions. The database consisted of two tables representing "objects" — persons and movies, and two tables that related persons to movies either as an actor or as someone who had a major role in the creation of the movie. Note that there could be multiple rows relating a person to a movie as either an actor or crew or both. SQL queries were written that would produce answers to common questions of who did what in a movie. Once the queries had been tested and produced reasonable results for test cases, the queries were rewritten as template strings that could have names, roles or jobs filled in by the translator and printed as queries to the database.

The set of all jobs that are stored in the database corresponds directly to the words that need to be recognized as verbs in the translator. This need not be a one-to-one

correspondence as a person may wear several hats in the production of a movie or an actor may play multiple roles. The query processor rules were written to treat both verb-form and noun-form queries as equivalent; for example "Who directed…?" and "Who was the director of…?" will result in the same query template being used.

It is possible that a sentence is syntactically correct according to the rules of the query processor, but nonetheless may not be meaningful in English. As an example, "Who die 1787?" is described as "ungrammatical, but meaningful as a request for information" (Johnson & Bryant, 1994, p 222). The tool accommodates such ungrammatical but meaningful queries.

The query processor has employed a simple mechanism for handling pronoun references. Every time a noun or noun phrase is encountered in the processing of a query, it is saved for use in the event a subsequent query uses a pronoun. The query processor makes no attempt to ensure that a pronoun it encounters makes sense in relation to the remembered noun. In the current implementation, no checks for gender or number are performed. This is an area that could be further explored to improve the conversational nature of the query processor.

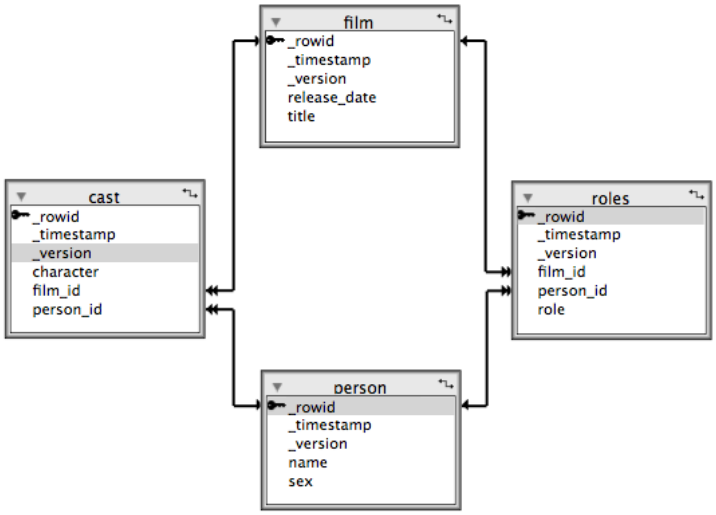
The enormity of the English language dictates a small subset of possible sentences be analyzed for translation into database queries. Additionally the "domain of inquiry" or the contents of the database to be queried needs to be narrowly focused. The attributes and relations in a database determine what questions can be asked of the database. Even with the simple movie database, complex queries that would require a logical connective between two attributes weren't examined. Any of the simple queries could be extended

into a complex query by the addition of a restrictive clause that referred to other attributes in the database.

4. Conclusions

There are several advantages to the approach discussed in this paper to building a natural language processor front end to a database: It uses tools that are robust, reliable and freely available. In choosing to use both a grammar-driven parser generator as the tool for creating a query processor and SQL as the target result of the translation, the work of hand-coding a parser and a database system is avoided. The programming was done using widely known languages (C and SQL.) Lastly, a person who uses this tool to query a database doesn’t need to learn a new language to do so.

Appendix



1. Database schema

```
ask away! Who played Luke Skywalker in "Star Wars"?
select
      p.name
            from film f, person p, cast c
            where
                c.character like "%Skywalker%"
            and p.row_id = c.person_id
            and f.row_id = c.film_id
            and f.Title like "%Star Wars%";
```

2. Conversation with the query processor.

```
openbase 1> select
openbase 2>  p.name
openbase 3>   from film f, person p, cast c
openbase 4>   where
openbase 5>       c.character like "%Skywalker%"
openbase 6>   and p.row_id = c.person_id
openbase 7>   and f.row_id = c.film_id
openbase 8>   and f.Title like "%Star Wars%"
openbase 9> go
Data returned... calculating column widths

Name
-------------
Mark Hamill
-------------
1 rows returned - 0.003 seconds (printed in 0.003 seconds)
```

3. Database processes the generated query.

References

Bobrow, D.G. (1968). Natural language input for a computer problem-solving system. In M. Minsky, (Ed.), *Semantic information processing*. Cambridge, Mass: MIT Press.

Codd, E.F. (1970). A Relational model of data for large shared data banks. *Communications of the ACM* 13(6): 377–387. doi:10.1145/362384.362685

Erickson, J. (2008, March). Natural language and database technology, Dr. Dobb's Journal. CMP Media. Retrieved February 8, 2009, from http://www.ddj.com/database/20690463

Flowers, T.H. (1983, July). The Design of Colossus. *Annals of the History of Computing, 5*(3) pp239-252. Retrieved February 8, 2009, from http://www.ivorcatt.com/47c.htm

Hallett, C. 92006) Generic querying of relational databases using natural language generation techniques. *Proceedings of the fourth international natural language conference*. Morristown, NJ: Association for Computational Linguistics. 95-102

Hendrix, G.G., Sacerdoti, E. D.. Sagalowicz, D., & Slocum, J, (1978, June). Developing a natural language interface to complex data. *ACM Transactions on Database Systems, 3(*2), 105-147. New York, NY: ACM. doi:10.1.1.17.3033

Johnson, R.D. & Bryant, B.R. (1994). A quick method for answering wh-questions in a natural language database interface. Proceedings of the 7th Florida artificial intelligence research symposium. 221-225. Retrieved March 8, 2009, from http://www.cis.uab.edu/bryant/papers/flairs94.pdf

Johnson, S.C. (1978). Yacc: Yet another compiler-compiler. In *UNIX programmer's manual.* Murray Hill, NJ: Bell Telephone Laboratories.

Oettinger, A. G. (1960). *Automatic language translation*. Cambridge: Harvard University Press.

Petrick, S.R. (1982). Theoretical/technical issues in natural language access to databases. *Proceedings of the 20th annual meeting on Association for Computational Linguistics*. Morristown, NJ: Association for Computational Linguistics

Reiss, S. (2003, December). Hope is a lousy defense. *Wired*, *11*(12). Retrieved February 8, 2009, from http://www.wired.com/wired/archive/11.12/billjoy.html

Sidner, C. (1979). Disambiguating references and interpreting sentence purpose in discourse. In P. Winston & R. Brown, *Artificial intelligence: An MIT perspective Vol. 1., Expert problem solving, natural language processing, intelligent computer coaches, representation and learning*. The MIT Press series in artificial intelligence. Cambridge, Mass: MIT Press.

Thompson, C. W., Ross, K. M., Tennant, H. R. & Saenz, R. M. (1983). Building usable menu-based natural language interface to databases. In Schkolnick, M. & Thanos, C, (Eds.), *Proceedings of the 9th international conference on very large data bases*. San Francisco, CA: Morgan Kaufmann. 43-55. doi:10.1.1.99.3690

Tomita, M., (1987, January-June). An efficient augmented-context-free parsing algorithm. *Computational Linguistics, 13*(1-2), 31-46. Retrieved June 4, 2009, from http://acl.ldc.upenn.edu/J/J87/J87-1004.pdf

Vassiliou, Y., Jarke, M., Stohr, E. A., Turner, J. A., & White, N. H. (1983, December). Natural language for database queries: A laboratory study. *MIS Quarterly 7*(4), 47-61. Retrieved march 24,2009, from http://www.jstor.org/stable/248746