Technical Perspective: Natural Language to SQL Translation by Iteratively Exploring a Middle Ground

Jeffrey F. Naughton
University of Wisconsin–Madison
Madison WI, U.S.A.

A fundamental question in data management is how relational database management systems (RDBMSs) should be queried. Ideally, the query interface should be powerful enough to express arbitrary queries, yet simple enough to learn that users require virtually no training. Natural language is an obvious and appealing approach – presumably most users already know at least one natural language and use it to "query" other humans constantly. Unfortunately, employing natural language to query RDBMSs is highly nontrivial, and for the most part, not used. However, with the growing power and ubiquity of Natural Language Processing (NLP) systems, it makes sense to redouble efforts in applying NLP to database querying.

At the most basic level, relational database systems are queried using SQL. (For that matter, most "NoSQL" systems are also queried using SQL.) SQL is very powerful and precise, and, for novices, very hard to write. So SQL cannot be used as a user interface for anyone but power users. Nonetheless, as the most widely used RDBMS query language, SQL is the most natural language into which to translate natural language questions over relational data. This translation is the focus of the following paper, "Understanding Natural Language Queries over Relational Databases", by Li and Jagadish.

The first important decision made by the authors of this paper is to reject a one-shot, one-way translation process from a natural language query to a corresponding SQL query. Instead, the authors advocate an iterative dialog between the person posing the query and the system building the relational query. This makes perfect sense – even in the much simpler world of keyword search systems, users iteratively refine their queries. Unfortunately, adopting this approach for RDBMS querying does not yield an easy problem – in fact, it uncovers a highly interesting and difficult challenge: how should the user and the system communicate in this iterative process?

Answering this question is difficult. Unlike the case for keyword search systems, the answer to the query may not help the user know if the executed query was what they really wanted. For example, consider the simple query "find the difference between sales this year and last year." In general the RDBMS will return a number – and it is very hard to tell just from that number if the query was correct or not. It would be far more precise for the system to respond to the user by presenting the generated SQL query itself. But this would require the person posing the natural language query to be able to read and understand SQL, which contradicts a major motivation for the system in the first place.

Now we come to what is perhaps the heart of this paper: the decision to adopt an intermediate language the authors call "Query Tree," a two-way domain-independent communication model allowing the user and system to understand one other. A query tree aids mapping a user query to its corresponding semantically correct SQL and translating a query plan to its corresponding natural language interpretation. The authors harness the schema knowledge, schema-driven similarity metrics, query tree reformulation and ranking to make the problem tractable for the system and the user.

The authors close with a user study evaluating the approach. The user study itself is interesting, including the aspect of using Chinese to convey the queries to the subjects instead of English to avoid bias through the phrasing in the query description (presumably the subjects already spoke Chinese!) The experiments show that the approach is best for simple to medium complexity queries.

This paper represent a significant improvement in the state of the art, and it is an ideal springboard for future advances. In an area as difficult and important as natural language querying of relational database systems, this is indeed a major contribution.