Fragment-Driven Natural Language Interaction with Databases

Christopher Baik
cjbaik@umich.edu
University of Michigan
Ann Arbor, MI

Retrieving data from a relational database is a challenge for non-technical users. While SQL is a formidable swiss army knife for database tasks, even its core functionality, such as the ability to join tables, remains opaque for many users. This challenge is exacerbated by the fact that many production databases have complex schemas. Consequently, several lines of research [3, 8, 9, 11] have focused on helping less-technical users access databases via a natural language interface to database (NLIDB).

Several NLIDBs [3, 8, 10] follow a common interaction model:

N1. The user issues a free-form natural language query (NLQ) describing their request.
N2. The system generates candidate SQL queries from the NLQ.
N3. The user selects one of the SQL queries (or an intermediate representation of the SQL [3]), or returns to **N1** if none of the SQL queries match the request.
N4. The system executes the SQL query and returns an answer.

One pitfall of this interaction model is in **N3**, where the user is expected to select the correct SQL query from a list of candidates. Given that one of our goals in developing an NLIDB is to assist non-technical users without knowledge of SQL, it is contradictory to expect the user to understand standalone SQL queries without any additional annotation during the interaction. In addition, several previous systems lack transparency in the translation process, whether by producing SQL queries with little indication as to how they were produced [8, 10, 11] or requiring users to understand complex intermediate representations [3].

We propose an alternative *fragment-driven interaction model*, where the system provides an explanation as to how the natural language produced the resulting SQL:

I1. The user issues a free-form NLQ describing their request.
I2. The system decomposes and rephrases the NLQ into natural language fragments (NLF) which each map to a portion of a generated SQL query.
I3. The user views the system interpretation and modifies their NLQ by removing NLFs or adding suggested ones, which also modifies the resulting SQL.
I4. The system executes the final SQL query and returns an answer.

This interaction model enables the user to interact with the system purely in natural language and to make *incremental modifications* to their resulting database query without having to learn any SQL. In addition, for users unfamiliar with SQL, transparently displaying the mappings from NLF to SQL can provide *confidence* in the resulting query.

Supporting this interaction model poses several research challenges. First, the NLFs we obtain should have *high coverage* over all (or a large subset of) possible NLQ to SQL tasks. Second, the mappings from these NLFs *unambiguous, high-quality mappings* to SQL should be as *unambiguous* as possible—i.e. one NLF should not refer to two different SQL fragments. Third, the system should be able to easily adapt to *new domains and databases*.

To solve these challenges, we propose a system named FragSQL, which leverages *natural language fragment templates* to model NLQ to SQL tasks, a *fragment template mining* algorithm to extract natural language fragments from existing NLQ to SQL datasets, and provides *explanations and suggested query modifications* to the user through the interface. In addition, FragSQL can be adapted to new domains and databases by providing a few domain-specific NLQ to SQL examples for each database.

FragSQL demands less user expertise than an approach [3] which requires the user to investigate natural language parse trees. Unlike previous natural language explanation approaches [1, 4–7], FragSQL does not require users or administrators with schema knowledge to manually create a translation table or knowledge base. It is also an improvement over [2], which generates explanations that flesh out the user's original NLQ with values from the database, but is unable to correct potentially flawed SQL interpretations.

REFERENCES

[1] H. Amano and Y. Kambayashi. Translation of sql queries containing nested predicated into pseudonatural language. In *DASFAA*, pages 116–125. World Scientific, 1991.
[2] D. Deutch, N. Frost, and A. Gilad. Provenance for natural language queries. *Proceedings of the VLDB Endowment*, 10(5):577–588, 2017.
[3] F. Li and H. Jagadish. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84, 2014.
[4] J. Ljungberg. Paraphrasing sql to natural language. In *Intelligent Text and Image Handling-Volume 2*, pages 790–808. Le Centre de Hautes Etudes Internationales d'Informatique Documentaire, 1991.
[5] B. G. Lowden and A. N. De Roeck. The remit system for paraphrasing relational query expressions into natural language. In *VLDB*, pages 365–371, 1986.
[6] W. Luk and S. Kloster. Elfs: English language from sql. *ACM Transactions on Database Systems (TODS)*, 11(4):447–472, 1986.
[7] E. M. Mueckstein and G. D. Moerdler. Semantic interpretation of a database query language. *Data & Knowledge Engineering*, 1(2):123–138, 1985.
[8] A.-M. Popescu, O. Etzioni, and H. Kautz. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157. ACM, 2003.
[9] D. Saha, A. Floratou, K. Sankaranarayanan, U. F. Minhas, A. R. Mittal, and F. Özcan. Athena: an ontology-driven system for natural language querying over relational data stores. *Proceedings of the VLDB Endowment*, 9(12):1209–1220, 2016.
[10] N. Yaghmazadeh, Y. Wang, I. Dillig, and T. Dillig. Sqlizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):63, 2017.
[11] T. Yu, M. Yasunaga, K. Yang, R. Zhang, D. Wang, Z. Li, and D. Radev. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663, 2018.