Towards Robust and Transparent Natural Language Interfaces for Databases

Christoph Brandt
TU Darmstadt

Benjamin Hättasch
TU Darmstadt

Nadja Geisler
TU Darmstadt

Carsten Binnig
TU Darmstadt

ABSTRACT

In recent years the field of research on natural language interfaces for databases (NLIDBs) has progressed considerably, which can be seen by the results of challenges like Spider. However, most of these approaches concentrate on delivering best-guess answers and improving (computational) accuracy. Yet, there are still a lot of open issues regarding robustness, confidence, and transparency. Therefore, this vision paper starts to point to relevant milestones as well as corresponding opportunities for addressing them, opening up a potential path of the future development of NLIDBs.

KEYWORDS

Natural Language Interfaces, SQL, Databases

ACM Reference Format:
Christoph Brandt, Benjamin Hättasch, Nadja Geisler, and Carsten Binnig. 2020. Towards Robust and Transparent Natural Language Interfaces for Databases. In *Workshop on Human-In-the-Loop Data Analytics (HILDA'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3398730.3399190

1 INTRODUCTION

Motivation: Natural language (NL) interfaces present a convenient way for non-technical users to retrieve information from databases. In recent years, black-box approaches based on deep learning have shown to dominate existing *unidirectional* rule-based or statistical approaches in providing natural language interfaces for databases (NLIDBs) [31]. However, there are still open issues in the sense that black-box approaches are not able to deliver explainable translations between natural language statements and SQL statements in a transparent way [7].

HILDA'20, June 14–19, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-8022-5/20/06…$15.00
https://doi.org/10.1145/3398730.3399190

For example, adversarial attacks of black-box approaches have shown that a slight modification of the input features of a deep model may lead to prediction errors with high confidence. For NLIDBs with black-box based translation approaches this might mean that they will translate the NL user query to a completely unrelated SQL query. This is a huge problem for users without technical knowledge since even when presented with the SQL translation they might not notice the problematic translation. This renders the black-box approaches unusable for critical domains such as medicine or finance where erroneous information can have a severe negative impact.

Contribution: In this vision paper, we outline what a more robust approach for NLIDBs could look like as a potential solution. At the core, we suggest the usage of rule-based translations. While rule-based approaches were common in the 1980s, our idea is different: First, we suggest using *bi-directional* rules that can provide a mapping from NL to SQL and back operating on abstract syntax graphs. With such a set of rules, any translation from NL to SQL could be translated back to NL to explain to the user what the system executed. Therefore, this approach differs from alternative solutions presented in [13] and [9]. Second, instead of manually crafting the rules, we suggest to automatically derive the rules using machine learning algorithms. The goal of this approach is to avoid the pitfalls of the systems developed in the 1980s by learning a large corpus of rules that can deal better with variety in the user input.

In the first part, we discuss the need for such a system by dissecting an existing natural language interface called DBPal [28] and reflect pitfalls shared among current systems. We present the results of a study showing the weaknesses of black-box approaches such as DBPal through some concrete examples that lead to erroneous translations. In addition to that, we identify the needs to cope with these pitfalls, sketch what kind of human-centered language features a potential solution should support, and suggest what kind of tool environment is required.

Outline: The remainder of this paper is organized as follows: In Section 2, today's pitfalls are reflected by looking into concrete issues of DBPal and identifying general issues such as missing extensibility and lack of language data. In

addition to that, critical needs are outlined by discussing robustness and transparency. In Section 3, a potential idea for a solution is discussed. Finally, we draw a conclusion in which we combine the pitfalls, needs, and the possible solution in Section 4.

2 TODAY'S PITFALLS AND NEEDS

In this section, today's pitfalls with regard to robustness and transparency of natural language interfaces for databases are reflected using DBPal[25; 28] as an example. DBPal requires a database schema in order to automatically generate a large collection of samples containing NL queries and their corresponding SQL statements based on a collection of seed templates. The pairs can then be used to train a black-box translation model.

2.1 Concrete Issues for the End-User

In the following subsection, some exemplary NL queries are introduced that lead to failures in DBPal. Although they are meaningful and could be asked by real-world users, the deep learning-based system[1] is not able to translate them correctly.

Working Questions. Let us start with a simple question that works. DBPal was trained for answering typical OLAP-style queries (i.e., aggregations). So, a query like (Q1) works fine.

```
Q1: "How many patients are in the system?"
```

DBPal answers that there are 100 patients in the system. It also provides the corresponding SQL query for this question.

```
1 SELECT count( 1 ) FROM patients
```

Simple Questions that fail. However, already using slight modifications of this simple question, which asks about female patients in a specific way (Q2), fail.

```
Q2: "How many female patients are in the system?"
```

For this query, DBPal returns the value 27 which seems reasonable but the executed query is unexpected:

```
1 SELECT count( 1 ) FROM patients WHERE patients.
    gender <= "female"
```

The main problem with this type of question is that the results seem to make sense for an end-user and thus they might not be able to identify that an erroneous translation happened.

Complex Questions that fail. While this already demonstrates that DBPal makes errors that are not easily identifiable by users, there are also other categories of problems:

For example, question (Q3) cannot be translated properly by DBPal (since such types of queries were not contained in the training set) but might make sense from a user perspective.

```
Q3: "Are there errors in the database?"
```

In this case, DBPal reports success and returns the value 100. It would, therefore, be plausible for the user to assume that this is a valid question, which was handled properly by the system. Yet unfortunately, the system resorted to counting the entries in the *patients* table and was not able to answer the intended question.

Insight and Final Thoughts. These examples shed some light onto the problem of why current black-box NLIDBs are not yet useful for end-users unfamiliar with SQL. In the following, we discuss two other problematic dimensions of black-box NLIDBs. In addition to the above, exploring potential boundaries of SQL queries, so that they can easily be mapped into NL statements, remains an open issue. The same holds for user interactions with query plans in this context [16].

2.2 Tomorrow's needs

Need for Explainability. One of the main problems of black-box NLIDBs is the missing explainability: it should be possible to show how a given statement in a domain language of the user space (e.g., a medical question) is mapped to a corresponding SQL statement for a given database schema in a detailed, step-by-step way, and vice versa.

This is an open issue for systems based on deep neural networks where transformations are realized atomically. Sequences of intermediate results of single transformation steps as they show up when working with bidirectional transformation rules like triple graph grammars (TGGs) are not available.

Need for Extensibility. Another dimension that makes it hard to use black-box approaches is missing extensibility, i.e. a possibility to easily adjust the translation system to changes in the database schema or the domain language. This is an open issue since state-of-the art systems are using trained neural networks where the database schema and the domain language are baked into the model. Changes require the network to be retrained with modified training data.

A major issue in this context is the scarcity of suitable data, in this case, pairs of NL and SQL statements. Unfortunately, this data is a) hard to obtain and b) even if there is data available, it is hard to evaluate if the amount and variety where enough for the trained system to generalize.

Need for Robustness. As discussed earlier, many more challenges arise when using NLIDBs in settings of safety-critical applications such as hospitals. In these environments, deployed solutions that come with natural language interfaces need to be robust and transparent in order to become eligible for insurance coverage:

[1] https://dbpal.dm.informatik.tu-darmstadt.de/

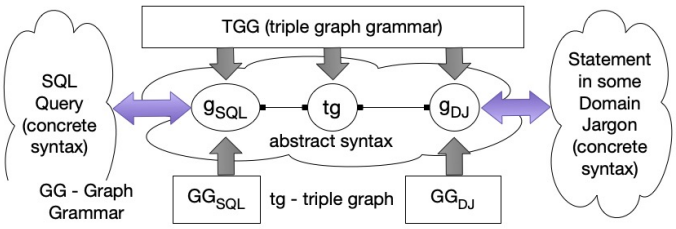

Figure 1: A TGG represents a set of rules that translate NL step-wise into a SQL statement and vice versa.

Concerning robustness, it should be possible to handle complex questions that require join and nest operations, requests entered as a set of keywords, grammatical alterations, variations in vocabulary and syntax, different linguistic styles of phrasing, ill-formed or syntactically incomplete questions, and questions that are characterized by missing information [28]. A potential opportunity concerns conversational agents between the NLIDB and the user to get feedback from the user or to offer options such as auto-completion. Furthermore, another opportunity is to support the constructive development of queries in a step-by-step way through interactions between the end-user and the system in a session-based dialogue [32][2].

Need for Transparency. Concerning transparency, it should be possible for end-users to trace down failure cases and correct mappings. This requires a) a way for the user to decide whether his information request was translated correctly by the NLIDB (e.g., through deterministic back-translation and query explanation) and b) a way to correct or specify their request (e.g., through iterative refinement [28] or a guided clarification process by a conversational agent[32]).

3 A POTENTIAL SOLUTION

The main idea of this approach is summarized in Figure 1. Initially, an NL question is translated into a graph representation using existing approaches for semantic parsing. Afterwards, we use triple graph grammars (TGGs) [23; 29] that translate the semantic parse tree into a SQL tree as a set of graph transformations [10].[3]

As mentioned at the beginning, we propose a solution where the set of translation rules (i.e., the TGG) is learned from a training set. To synthesize training data for learning the rules we need NL/SQL pairs. We propose to use a data generation approach such as the one used by DBPal. More details about the learning process are discussed in Section 4. In the following, we discuss some basic properties of rules expressed as a TGG which make them an interesting alternative to encode rules for translating NL to SQL.

Support for bidirectional Alignment. The grammar of a language like SQL can be used to generate valid statements whose abstract syntax trees are encoded as graphs, whereas its grammar can be represented as a graph grammar. In contrast to that, any kind of domain jargon (DJ) comes with a set of given statements whose abstract syntax trees are represented as graphs, but that may lack a fully specified corresponding grammar. TGGs are especially interesting since they support a bidirectional mapping of graphs for those languages and thus can not only be used for the translation but also for back-translation to explain SQL to the user as an NL question [4; 5].

Specific contribution: Because TGGs are bidirectional, the same rule set is able to support the mapping from a statement expressed in DJ to a SQL query and vice versa. Therefore, there is no need to create a separate implementation for each direction.

Support for Evolution: As discussed before, the set of questions in NL/DJ a system needs be able to translate should be extensible. That is why a potential solution cannot assume a fixed input language specification of a DJ. Another benefit of TGGs is that the translation rules can be extended incrementally without violating correctness of the translation if the natural language needs to be extended. First results of how to evolve the NL artifacts whose abstract syntax is stored using RDF with graph transformation rules are discussed in [3].

Specific contribution: Rules of a TGG are fully associative. There-fore, rules sets formulated as a TGG can be extended with new rules easily because there is no specific order in which rules need to be applied.

Support for Incompleteness: Since any DJ as a subset of a natural language can also be incomplete and has subtle variations depending on the (group of) users, a potential solution cannot assume a fully specified version of the user's language. It needs to support the use of placeholders, like abstract nodes

[2] https://yale-lily.github.io/cosql

[3] The general idea of TGGs is to specify a language of integrated models that consist of a source and a target model as well as a correspondance structure in order to support bi-directional model transformations using graphs.

in a graph, representing an abstract syntax tree that might be replaced later on by concrete nodes when applying graph transformation rules, once the learning of the specific DJ has happened.

Specific contribution: Since TGGs might encompass abstract nodes, incomplete parts of a language can be modeled as part of a TGG. Different from black-box models, TGGs shall be able to define the variability of the input language explicitly.

Support for Ambiguity: In contrast to machine-cen-tered languages, natural languages are inherently ambiguous, which turns out to be a strong feature of these languages, not a defect. Since this also holds for any DJ as a subset of a natural language, a potential solution that maps statements expressed in a DJ into, e.g., SQL statements, needs to get access to some kind of context. This can be used to resolve ambiguity issues by integrating the graph representing the abstract syntax tree of statements encoding end-user queries with the graph representing the abstract syntax tree of statements encoding knowledge using triple graph rules. First results of how to handle language ambiguities already have been discussed in [15] using re-construction techniques.

Specific contribution: TGGs also support the alignment of statements in a DJ encoding the query and the available context to resolve ambiguities.

Support for Composability: Because a natural language interface is a result of a software-engineering process (see also [27]), the same holds for its components. As such, the specification of a DJ used by an NL interface is also an engineering result, and because within any constructive engineering process there is a tendency to compose new solutions out of already available components, a potential solution needs to support the engineering of languages as discussed in [1; 11; 21] by composing already available building blocks, whereas theoretical support for composability comes from category theory as discussed in [24].

Specific contribution: TGGs rule sets shall be composable, like languages are composable.

Support for Agility: Because, as discussed above, a language artifact expressed in a specific DJ can be incomplete or ambiguous, a potential solution needs to support dialogues in order to come back to the user and ask for advice of how to understand a certain statement. In addition to that, humans like to use agile protocols that are not preventing, for example, spontaneous meta-questions as it is the case with strict technical protocols as discussed in [17] and in [6]. Therefore, a potential solution needs to support agile dialogues. That could be specified applying a compositional approach of constructing language models that not only talk about the syntax and semantics of the language, but also about its pragmatics. This is meant to be the operational semantics of speech-acts as part of agile protocols expressed in that language as discussed in [6] with support for composability from category theory [24].

Specific contribution: TGGs rule sets shall be compatible with embedded languages created through composition.

4 MISSING PIECES

As discussed before, we believe that TGGs are an interesting direction to develop NLIDBs. However, developing TGGs for NLIDBs is not trivial. In the following, we discuss the missing pieces needed to bootstrap a TGG-based NLIDB:

Learning Rules: Since handcrafted TGG rule sets are cumbersome when they reach a certain size, machine learning approaches could be applied to derive those rules [26]. The learning of grammar rules has been discussed in [2; 8; 12]. It might help to combine classical rule-driven approaches from the 1970s-1990s with machine learning approaches from 2014-today as discussed in [19; 26]. As long as such rule sets are composable, they can easily be extended by additional rules. Once an alignment by rules between statements expressed in, e.g., SQL and a certain DJ (and vice versa) is established, a transparent mapping becomes available naturally.

Iterative Development: In case that language evolution and composition is required, design science [30] and constructivism [18] might be very helpful methods [20]. Design science could be able to support the iterative development of rule-based mutual alignments, whereas constructivism supports the composition of languages out of predefined elements.

Dialog-based NLIDBs: Robustness could be achieved using dialog-based approaches for context clarification. As a consequence, not only the syntax and semantics of a language need to be engineered to be able to express statements, but also its pragmatics to support dialogues. In addition to that, context may help to resolve ambiguities or fix incomplete statements, which require certain domain knowledge to be engineered in the context of an already engineered language [14; 19; 22].

ACKNOWLEDGEMENTS

We gratefully acknowledge the support of the members of the KIDS Group at Oracle (Matthias Brantner, Dieter Gawlick, Kirk Bradley, Kenny Gross, Anna Chystiakova, Greg Pogossiants, Paul Sonderegger and Zhen Hua Liu) for providing their extensive feedback and experience and the financial support of the External Research Office (ERO) at Oracle Labs.

REFERENCES

[1] MPS: The Domain-Specific Language Creator by JetBrains. Library Catalog: www.jetbrains.com.

[2] K. Ates et al. Graph Grammar Induction on Structural Data for Visual Programming. In *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, pages 232–242, Nov. 2006. ISSN: 2375-0197.

[3] B. Braatz et al. A framework for families of domain-specific modelling languages. *Software and Systems Modeling (SoSyM)*, 13(1):109–132, Feb. 2014.

[4] C. Brandt et al. Generation and Evaluation of Business Continuity Processes; Using Algebraic Graph Transformation and the mCRL2 Process Algebra. *Journal of Research and Practice in Information Technology*, 2010. Place: Sydney, New South Wales, Australia Publisher: Australian Computer Society Inc.

[5] C. Brandt et al. How Far Can Enterprise Modeling for Banking Be Supported by Graph Transformation? In H. Ehrig, A. Rensink, G. Rozenberg, and A. Schürr, editors, *Int. Conf. on Graph Transformation (ICGT 2010)*, volume 6372 of *Lecture Notes in Computer Science*, pages 3–26. Springer, 2010.

[6] C. Brandt et al. Syntax, Semantics and Pragmatics in Communication. In *7th International Conference on Semantic Systems*, Messe Congress Graz, Austria, Sept. 2011. ACM.

[7] C. Brandt et al. A modern approach to situation awareness: The ultimate challenge for event processing. In *Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems, DEBS 2019, Darmstadt, Germany, June 24-28, 2019*, pages 193–196. ACM, 2019.

[8] C. de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.

[9] D. Deutch, N. Frost, and A. Gilad. Explaining natural language query results. *VLDB J.*, 29(1):485–508, 2020.

[10] H. Ehrig et al. *Graph and Model Transformation: General Framework and Applications*. Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin Heidelberg, 2015.

[11] M. Folwer. Language Workbenches: The Killer-App for Domain Specific Languages? Library Catalog: martinfowler.com.

[12] L. Fürst et al. Graph Grammar Induction as a Parser-Controlled Heuristic Search Process. In A. Schürr et al., editors, *Applications of Graph Transformations with Industrial Relevance*, Lecture Notes in Computer Science, pages 121–136, Berlin, Heidelberg, 2012. Springer.

[13] G. Koutrika, A. Simitsis, and Y. E. Ioannidis. Explaining structured queries in natural language. In F. Li, M. M. Moro, S. Ghandeharizadeh, J. R. Haritsa, G. Weikum, M. J. Carey, F. Casati, E. Y. Chang, I. Manolescu, S. Mehrotra, U. Dayal, and V. J. Tsotras, editors, *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*, pages 333–344. IEEE Computer Society, 2010.

[14] T. Kuhn. *Controlled English for Knowledge Representation*. PhD thesis, Faculty of Economics, Business Administration and Information Technology of the University of Zurich, 2010.

[15] F. R. Lehmann. *Fachlicher Entwurf von Workflow-Management-Anwendungen*. Teubner-Reihe Wirtschaftsinformatik. Teubner, 1999.

[16] S. Liu, S. S. Bhowmick, W. Zhang, S. Wang, W. Huang, and S. Joty. NEURON: Query Execution Plan Meets Natural Language Processing For Augmenting DB Education. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, pages 1953–1956, Amsterdam, Netherlands, June 2019. Association for Computing Machinery.

[17] B. Meyer. *Agile! The Good, the Hype and the Ugly*. Springer Publishing Company, Incorporated, 2014.

[18] J. Mittelstrass. *Enzyklopädie: Philosophie und Wissenschaftstheorie, Band 1-4*. J.B. Metzler, Stuttgart, Germany, 1995. Sonderausgabe.

[19] G. D. Németh. Machine Translation: A Short Overview, Oct. 2019. Library Catalog: towardsdatascience.com.

[20] E. Ortner. Wissensmanagement, teil 1: Rekonstruktion des anwendungswissens. *Informatik Spektrum*, 23(2):100–108, 2000.

[21] E. Ortner. Wissensmanagement, teil 2: Systeme und werkzeuge. *Informatik Spektrum*, 23(3):192–201, 2000.

[22] E. Ortner. Sprachingenieurwesen empfehlung zur inhaltlichen weiterentwicklung der (wirtschafts-)informatik. *Informatik Spektrum*, 25(1):39–51, 2002.

[23] A. Schürr et al. 15 Years of Triple Graph Grammars. In H. Ehrig, R. Heckel, G. Rozenberg, and G. Taentzer, editors, *Graph Transformations*, pages 411–425, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[24] D. I. Spivak. *Category Theory for the Sciences*. The MIT Press, 2014.

[25] K. Stockinger. The Rise of Natural Language Interfaces to Databases – ACM SIGMOD Blog, 13-03-2020. Library Catalog: wp.sigmod.org.

[26] D. Torregrosa, N. Pasricha, M. Masoud, B. R. Chakravarthi, J. Alonso, N. Casas, and M. Arcan. Leveraging Rule-Based Machine Translation Knowledge for Under-Resourced Neural Machine Translation Models. In *Proceedings of Machine Translation Summit XVII Volume 2: Translator, Project and User Tracks*, pages 125–133, Dublin, Ireland, Aug. 2019. European Association for Machine Translation.

[27] R. Turner and N. Angius. The philosophy of computer science. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2020 edition, 2020.

[28] P. Utama et al. An End-to-end Neural Natural Language Interface for Databases. *arXiv:1804.00401 [cs]*, Apr. 2018. arXiv: 1804.00401.

[29] N. Weidmann et al. Incremental bidirectional model transformation with emoflon::ibex. In *8th International Workshop on Bidirectional Transformations*, pages 45–55, 2019.

[30] R. Winter et al. Restructuring the Design Science Research Knowledge Base. In R. Baskerville, M. De Marco, and P. Spagnoletti, editors, *Designing Organizational Systems*, volume 1 of *Lecture Notes in Information Systems and Organisation*, pages 63–81. Springer Berlin Heidelberg, 2013.

[31] T. Yu et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921, 2018.

[32] T. Yu et al. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases, 2019.