Let the Database Talk Back: Natural Language Explanations for SQL

Stavroula Eleftherakis
Athena Research Center
Athens, Greece
seleftheraki@athenarc.gr

Orest Gkini
Athena Research Center
Athens, Greece
orestg@athenarc.gr

Georgia Koutrika
Athena Research Center
Athens, Greece
georgia@athenarc.gr

ABSTRACT

Database interaction is often characterized as a non-trivial and time-consuming process due to user's inexperience with the data or the query language. Therefore, there is a need for the databases to be able to "talk back" in order to assist the users during data exploration and eventually lead them to the desired results. In this paper, we tackle the problem of SQL-to-NL by extending the graph-based model of Logos [3]. Our novel extensions include improvements in terms of the system's translation capabilities and the fluency of the generated explanations. Finally, we report several challenges, highlighted by experiments on different user cases, i.e, astronomy and policy making.

Reference Format:
Stavroula Eleftherakis, Orest Gkini, and Georgia Koutrika. Let the Database Talk Back: Natural Language Explanations for SQL. In the 2nd Workshop on Search, Exploration, and Analysis in Heterogeneous Datastores (SEA Data 2021).

1 INTRODUCTION

Nowadays, the availability of data coming from many and different application fields urges the deployment of sophisticated tools which help us effectively explore and extract knowledge out of this data. However, the variety of the data sources (astronomical, biomedical, etc.) as well as the complexity of query languages such as SQL and SPARQL often pose obstacles during data exploration due to the users' unfamiliarity with the database content or the query language. Explaining queries in text can help tackle both problems as it enables users to understand the SQL queries that are used to retrieve the answers through the data exploration process [8].

The problem of translating SQL queries to natural language, or SQL-to-NL in short, appears to be deceivingly simple, as SQL queries are using a restricted vocabulary, comprising SQL elements (clauses, operators, etc.) and database elements (i.e., relations, attributes), and there is no ambiguity in interpreting such elements. Hence, initially a straightforward translation appears adequate. Reality teaches us quite the opposite, as the resulting text should be accurate in capturing the respective SQL query, and effective allowing fast and unique interpretation of it. Achieving both of these qualities is very difficult and raises several technical challenges that need to be addressed [8]. Furthermore, several different explanations can be generated for the same SQL query, making evaluation challenging.

Logos [3] is a well-established system which translates SQL queries into narratives. In contrast to neural approaches, Logos' model can be interpreted. This gives us the advantage of offering precise translations, knowing exactly how translations are produced. Nevertheless, the system's translation capabilities are highly depended on the query's syntax, meaning that for each type of query the system should have the necessary tools to model it. Moreover, depended on the query's complexity (number of attributes, bridge tables, etc.) translations can become unnatural.

In our work, we provide NL explanations as part of a data exploration platform used by our collaborators representing different fields, including astrophysics and policy making [1]. In this platform, NL explanations are provided for explaining the internal SQL queries generated by different system components used for recommendations, NL-to-SQL translation, and data exploration. Thus, we have to translate a variety of queries containing different clauses and operators. Furthermore, there is a need for these translations to be as natural is possible.

In an effort to tackle the above-mentioned challenges, we extend Logos to two important directions: (a) *translation capabilities*, where we focus on the system's ability of translating different types of queries, (b) *fluent explanations*, where we focus on improving the system's translations. Furthermore, we present *experimental results as well as results of a user study* over two different databases: astronomical data and policy-making data. Our evaluation shows the effectiveness of our approach and provides several insights regarding challenges that arise due to the ambiguous nature of the NL explanations including: (a) scoring textual explanations, and (b) generating explanations suitable for different groups of people.

The rest of the paper is organised as follows. In Section 2, related work is discussed. In Section 3, we provide background information on Logos. In Section 4, we present our novel extensions. In Section 5, the effectiveness of the system is explored. Finally, concluding remarks are provided in Section 6.

2 RELATED WORK

Existing approaches for SQL-to-NL can be broadly divided into two categories. *Template-/rule-based approaches* (e.g., [4], [5]) require the design of special templates and rules that are used to compose sentences. Due to the non-ambiguous nature of the SQL queries, templates can help limit the syntactic variability of their output and smooth explanations out [7]. Logos [3] falls into this category.

The second line of research tackles the problem as *neural machine translation* and uses sequence-to-sequence (Seq2Seq) models (e.g., [2], [12]) which treat both the natural language description and

Published in the Proceedings of the 2nd Workshop on Search, Exploration, and Analysis in Heterogeneous Datastores, co-located with VLDB 2021 (August 16-20, 2021, Copenhagen, Denmark) on CEUR-WS.org.

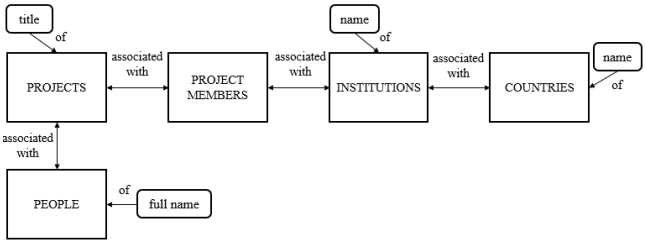

Figure 1: A subgraph of the CORDIS database graph.

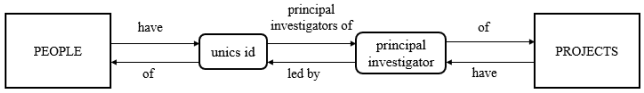

Figure 2: A join on the CORDIS database graph.

the query as sequences. In contrast to the previous category, those models automatically learn how to translate queries without the need of predefined query patterns. However, *they require plenty of training data and fine tuning and their effectiveness is still very low.*

Lately, hybrid models (e.g., [10], [11]) have been proposed to solve a similar problem to ours, i.e., data to natural language. Those models automatically learn templates and use them in order generate textual explanations.

3 BACKGROUND

Logos' approach to NL generation comprises graph representations of the database and the queries, template phrases associated with parts of the graph, and graph traversal in particular directions to compose the templates found on the way into the final text formation. We provide an overview of the main ingredients: the database graph, the query graph, labels, templates, and the algorithms.

The *database graph* captures the relationships between the database relations and attributes (nodes of the graph). Its edges are divided into three types: (a) *membership edges* (attributes to relations), representing attribute projections in queries, (b) *selection edges* (relations to attributes), representing attributes in predicates, and (c) *join edges* (attribute to attribute), representing joins.

The *query graph* of a query is the part of the database graph that the query refers to, extended with additional nodes (e.g., for functions) and edges (e.g., for group-by's) in order to capture the entire query meaning. Each node or edge of the query graph can be annotated with a *label* that signifies its meaning in natural language. There are default labels that can be used for any database. Additionally, labels can be provided by a domain expert.

Example. Let us consider the CORDIS [1] database, which stores information about research projects funded by the European Union. In Figure 1, we see a subgraph of the corresponding database graph. Note that for simplicity, we do not show the attributes used to join the various relations. By default, the name of each node is also its label. We also see how edges are annotated with default labels. We can easily override these. For instance, for the relation PROJECT_MEMBERS, instead of the system's default label *"project members"*, we can use the short label *"participants"*.

[1] https://data.europa.eu/euodp/en/data/dataset/cordisH2020projects

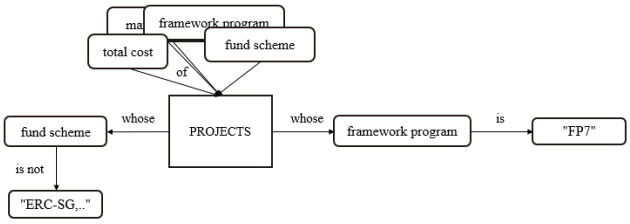

Figure 3: A query with IN and NOT IN operators.

Figure 2 zooms in on the join path connecting relation PEOPLE with relation PROJECTS and provides a more detailed view. A possible label which expresses this connection is *"principal investigators of"*, while the default one is *"associated with"*. Such designer labels are stored in special tables called *designer tables*. □

NL explanations are created by traversing the query graph accompanied with a *template mechanism* [9]. The template mechanism uses the provided labels (designer or default) to form meaningful phrases. A *template label* $l(u)$ or $l((u,v))$ is assigned to a node u or a path (u,v), respectively. For instance, a join template label may have the form: $l((u,v)) = l(u) + expr + l(u,v) + l(v)$, where u and v are relation nodes of the query graph. Using the example of the previous paragraph (see Figure 2), a template for the path connecting the tables PEOPLE and PROJECTS is the following: $l(\text{(PEOPLE, PROJECTS)}) = l(\text{PEOPLE}) + \text{" as "} + \text{"principal investigators of"} + l(\text{PROJECTS}) = \text{"people as principal investigators of projects"}$.

Synthesis of the NL explanations is performed as graph traversal of the query graph. There are three traversal strategies: (a) Binary Search Tree (BST) algorithm, where the translation consists of a composition of clauses each one of them focusing on specific query semantics, (b) Multiple Reference Points (MRP) algorithm, where information from all parts of the query graph is blended in the translation, (c) Template composition (TMT) algorithm, where predefined, richer templates corresponding to different query parts are used in an effort to produce more concise translations. In what follows, our examples were created using the MRP traversal strategy and correspond to queries targeting the CORDIS database.

4 TOWARDS RICHER TRANSLATIONS

Logos was extended in two different directions. The first direction aimed to enable the translation of more query types, while the other one aimed to create fluent explanations. For this purpose, we have upgraded the query parsing, graph generation, traversal strategies, and template mechanism.

Translation Capabilities. Our first goal was to extend the translation capabilities of Logos towards translating more clauses and operators. To that end, Logos should be able to *analyze* those new types of queries, *enrich* (if necessary) the query graph with new types of nodes and edges, and *translate* those new elements using any of the traversal strategies (BST, MRP, TMT). Thus, we have implemented changes in the system's parser, introduced new graph elements, and created new translation rules. As a result, the system is now capable of translating queries with the `SELECT TOP` and `LIMIT` clauses, as well as the `(NOT) IN` and `(NOT) LIKE` operators.

Once an input query is given to the system, the parser analyzes it and generates a *parsing tree* that stores important information for

Table 1: Example SQL queries and generated interpretations.

SQL Query Interpretations
1. SELECT TOP 10 title FROM projects WHERE start_year=2021;
"Find projects whose start year is 2021. Limit the results to top 10."
2. SELECT total_cost, ec_max_contribution, framework_program, ec_fund_scheme FROM projects WHERE framework_program IN ('FP7') AND ec_fund_scheme NOT IN ('ERC-SG', 'ERC-CG');
"Find the total costs, max contributions, framework programs and fund schemes of projects whose framework program is FP7 and fund scheme is not ERC-SG and ERC-CG."
3. SELECT title FROM topics WHERE title LIKE '%climate%';
"Find the titles of topics whose title is like %climate%."
4. SELECT COUNT(∗) FROM projects WHERE start_year=2012;
"Find the number of projects whose start year is 2012."

the creation of the query graph. For the parser, we have introduced to the system several new parsing nodes (e.g., for limit) capturing information about all the aforementioned clauses and operators.

Queries having `SELECT TOP` or `LIMIT` clauses are marked as "limited" and their associated limitation number is temporally stored in the system. The translation algorithm (BST, MRP, TMT) checks if the input query is limited and if that is the case an explanation is separately created and added to the end of the translation. As an example, we provide that of query 1 in Table 1.

In order for Logos to translate queries having `IN` and `NOT IN` operators, new types of query graph elements were created: the *in edge*, the *not-in edge*, and the *value-list* node. An illustrative example of this case is query 2 in Table 1 whose query graph is given in Figure 3. The translation algorithm detects those new elements of the query graph and produces a textual explanation.

Regarding the translation of queries having the `(NOT) LIKE` operator, we created two new types of edges: the *like edge* and the *not-like* edge. An example of this case is that of query 3 in Table 1.

Our last improvement, in terms of translation capabilities, is the creation of a new query graph node, the *star node*. In combination with the function node that represents the `COUNT` operator, we now get explanations like that of query 4 in Table 1.

Fluent Explanations. Our second goal was to improve the translations generated by the system in terms of fluency. This kind of changes require only the modification of the traversal strategies and the template mechanism.

While exploring the system's capabilities, we noticed that many database schemas include bridge tables, i.e., special tables used to transform many-to-many relationships into one-to-many relationships. Bridge tables (usually) appear in queries in order to join two tables. Although those tables play an important role in terms of data modeling, they cause "noisy" explanations. For instance, let us consider query 1 of Table 2. Apparently, the previously produced explanation of the system is unnatural. Bridge tables are manually stored in a designer table and excluded from the translation. The template mechanism has been modified so that if u, v, and w are table nodes with v corresponding to a bridge table, the template label of the path connecting u to w is of the form $l((u, w)) = l(u) + l(u, w) + l(w)$.

Moreover, a mini dictionary has been developed providing the translation process with the plural form of all the attribute labels.

Table 2: Example SQL queries and generated interpretations, where B and A denotes before and after, respectively.

SQL Query Interpretations
1. SELECT pr.* FROM projects pr, project_subject_areas psa, subject_areas sa WHERE pr.unics_id = psa.project AND psa.subject_area = sa.code AND sa.title = 'Mathematics and Statistics';
B: *"Find everything about projects associated with project subject areas, and for project subject areas associated with subject areas whose title is Mathematics and Statistics."*
A: *"Find everything about projects on subject areas whose title is Mathematics and Statistics."*
2. SELECT COUNT(p.title), t.title FROM topics t, project_topics pt, projects p WHERE t.code = pt.topic AND pt.project = p.unics_id GROUP BY t.title;
B: *"Create groups according to the title of topics. Find the number of projects and the title of topics associated with project topics associated with the projects."*
A: *"Find the titles of topics and the number of projects on these topics grouped by the titles of topics."*
3. SELECT m.member_name FROM project_members m, projects p WHERE m.project = p.unics_id;
B: *"Find the member name of project members associated with projects."*
A: *"Find participants participated in projects."*

Essentially, for every attribute node u of the query graph we take (if that exists) the plural form p of its label $l(u)$, i.e., $p(l(u))$.

In addition, we have modified the way the MRP algorithm translates the group by clause. Previously the translation for this part would result in a separate sentence. Leveraging the nature of the algorithm which aims on generating translations by blending different parts of the query, we now blend this translation more naturally.

An example that captures all the aforementioned changes (bridge tables, plural form, group by on MRP) is that of query 2 in Table 2.

Lastly, we have improved the translation of queries which include only *heading* attributes [4] in the `SELECT` clause. The heading attribute is the most characteristic attribute of its relation. As an example we offer that of query 3 in Table 2.

5 EXPERIMENTAL RESULTS

The evaluation of Logos is divided into two parts: (a) the automated evaluation part, where we evaluate our results using the Bilingual Evaluation Understudy (BLEU) automated metric [6], and (b) the human evaluation part, where we evaluate our results using the help of SQL experts. The purpose of the first part is to use a well-established metric to show how good the NL explanations are, while the second part aims at evaluating qualitative aspects of the NL explanations, such as clarity and fluency.

For both types of evaluation, we created 28 queries (14 for the CORDIS database, and 14 for the SDSS [2] database) (see Tables 6 and 7, respectively). Using the MRP algorithm, we translated those queries twice. One time using the default version of the system, which works by considering only the database schema (internal knowledge), and one time using both the database schema and the designer tables (internal and external knowledge), which, as mentioned in the previous sections, store information about heading

[2] https://www.sdss.org/

Table 3: BLEU scores for the textual explanations of the 14 CORDIS queries, and the 14 SDSS queries.

Query ID	*CORDIS BLEU Scores*		*SDSS BLEU Scores*	
	Logos v.1	Logos v.2	Logos v.1	Logos v.2
1	3.21	3.98	4.99	12.55
2	4.37	6.27	2.45	2.66
3	2.26	9.26	11.71	15.73
4	3.51	12.87	4.86	5.01
5	2.01	15.46	2.84	3.67
6	3.40	11.20	3.38	4.07
7	4.03	4.32	3.74	4.37
8	8.23	9.55	14.72	14.01
9	12.30	9.29	4.46	18.80
10	18.30	7.41	4.03	4.07
11	14.46	24.81	18.46	27.36
12	10.70	17.40	6.87	8.56
13	4.07	4.20	4.30	22.24
14	3.67	4.46	35.61	5.50
MAX	18.30	24.81	35.61	27.36
MIN	2.01	4.20	2.45	2.66
MEDIAN	4.05	9.28	4.66	7.03

attributes, bridge tables, and node or edge labels. First of all, we want to investigate the effect of the designer tables to the translations. Moreover, we want to know how close to the ground truth (textual explanations given by SQL experts of the databases) the system's explanations are. In what follows, we denote the default version of the system as *Logos v.1* and the version that takes the advantage of the designer tables as *Logos v.2*.

5.1 Automated Evaluation

Automated evaluation was carried out to compare the generated explanations to the ground truth i.e., textual explanations of SQL queries given by SQL experts of the databases, members of the INODE project [1]. The quality of the results is measured using the BLEU-4 score. BLEU is a score for comparing a candidate translation of text to one or more reference translations. The scores range between 0% and 100%. A score of 100% means that the estimated, by the system, explanation matches completely the ground truth.

The results are summarized in Table 3. We also report the minimum and the maximum BLEU score. Furthermore, we noticed a large variation between the scores; thus, we decided to report the median BLEU score per translation system instead of the average. For both databases, median scores are under 10%. Looking at the medians, we conclude that Logos v.2 produces translations closer to the ground truth than those obtained from Logos v.1. Especially for the CORDIS database, the median BLEU score of Logos v.2 is more than two times higher than that of Logos v.1.

The low scores do not indicate that the translations produced by Logos are not correct. BLEU scores work by counting matching n-grams in the candidate translation to n-grams in the reference text, where 1-gram or unigram would be each token and a bigram comparison would be each word pair. That means that the score is higher the more common parts a NL explanation has with the ground truth. Manual examination of the NL explanations that the two versions of Logos generated versus the ground truth showed that the automatically generated translations were in fact correct. However, they looked very different from the ground truth. Indicatively, we show the translations of query 9 (from Table 6).

CORDIS query 9 explanations:

- **Logos v.1**: *"Find the institutions names of institutions associated with countries whose country name is France."*
- **Logos v.2**: *"Find institutions located in countries whose name is France."*
- **Ground truth**: *"Show names of institutions from France."*.

We see that Logos would not necessarily produce translations the way that a human mind would produce. And even different people would provide different explanations for the same SQL query (albeit all correct). This shows the opportunity of enhancing the translation capabilities of the system with learning that not only leverages the database schema but is also performed on previously defined human translations. It also shows the challenge of creating automated evaluation metrics which do not judge the explanations quality only by estimating their similarity with the ground truth. These observations lead to the need of conducting human evaluation as well, which will be presented in the next subsection.

Focusing now on the results of CORDIS, we see that there is significant improvement on the translations of queries with id 3-6, 11, and 12. This is mainly due to the exclusion of bridge tables from the translation procedure and the heading attribute addition. For queries with id 9, and 10 we noticed a score reduction. Indicatively, looking at the translations of query 9 above, we observe that although the translation of Logos v.2 is more natural than that of Logos v.1, the presence of the sentence "names of institutions" in the translation of the latter leads to a higher BLEU score.

Let us now focus on the results of SDSS. The scores are lower than those of the CORDIS database. This is due to the nature of the SDSS database that uses abbreviated names and letter symbols in order to describe the content of its tables and attributes. For instance, "photoobj" instead of "photometric objects", or the letter "u" to denote the magnitude of a photometric object in "u" (ultraviolet) filter. During the experiments, we realized that by transforming those names and symbols into meaningful textual sentences, we increase the size of the explanations compared to the size of explanations provided by the astrophysicist SQL expert. This shows another challenge for the automatic generation of NL explanations: different styles of explanations may be given by domain experts in different fields. Thus, different explanations are suitable for different groups of people (e.g., astronomers and data scientists). For example, for query 14, the BLEU score of Logos v.2 is substantially lower than that of Logos v.1.

SDSS query 14 explanations:

- **Logos v.1**: *"Find the u, g, r, i and z of photoobj associated with specobj whose class is QSO."*
- **Logos v.2**: *"Find the magnitude u, magnitude g, magnitude r, magnitude i and magnitude z of photometric objects corresponding to spectroscopic objects whose class is QSO."*
- **Ground truth**: *"Show me the u, g, r, i, z magnitudes of spectroscopic quasars."*.

We concluded that this kind of notation (abbreviated names, and letter symbols), for the attributes of the SDSS tables, is sometimes preferred over full descriptions.

Table 4: Average clarity, fluency, and precision per database, and translation system (their standard deviation in parentheses).

Features	*CORDIS*			*SDSS*		
	Logos v.1	Logos v.2	Ground truth	Logos v.1	Logos v.2	Ground truth
Clarity	4.25 (1.29)	5.79 (1.13)	6.79 (0.31)	4.32 (1.11)	6.04 (0.77)	6.50 (0.57)
Fluency	3.64 (1.43)	4.75 (1.06)	6.86 (0.35)	3.39 (0.97)	5.50 (0.87)	6.57 (0.56)
Precision	6.04 (1.46)	5.79 (1.10)	5.18 (1.75)	6.50 (0.50)	6.21 (0.70)	5.18 (1.01)

Lastly, it has been observed that the SDSS database includes many discrete variables (attributes) that define different types of objects, e.g., stars. A fine example of that case is that of query 7.

SDSS query 7 (Table 7) explanations:

- **Logos v.1**: *"Find the specobjids of specobj whose subclass is OB and class is STAR."*
- **Logos v.2**: *"Find spectroscopic objects whose spectroscopic subclass is OB and class is STAR."*
- **Ground truth**: *"Find all spectroscopic stars which are massive and hot."*.

We see that both versions of Logos do not understand that "subclass = OB" and "class = STAR" means massive and hot stars. This justifies the low BLEU scores in both versions of the system. For this purpose, additional knowledge is required. We are in fact in the process of integrating an ontology that provides such mappings.

5.2 Human Evaluation

For this experimental setting, an online survey was conducted. A total of 21 people, all SQL experts, participated in the survey. The experts were members of the INODE project [1]. People who contributed to the creation of labels (ground truth explanation) were excluded from the evaluation process. From the pool of 28 SQL queries (see Tables 6 and 7), participants were asked to rate the textual explanations of 4 randomly chosen queries (2 per database). The queries were equally distributed to all participants. For each query, the participant rated 3 explanations (1 per translation): (a) the explanation produced by Logos v.1, (b) the explanation produced by Logos v.2, and (c) the ground truth explanation, resulting in a total of 84 explanations rated by humans.

The participants judged the quality of the translations on the 7-point Likert-scales. They provided ratings for:

- *clarity*: how clear and understandable the explanation is.
- *fluency*: how natural the explanation is.
- *precision*: how well the information of the provided SQL query is captured on its textual explanation.

A score of 7 to all the aforementioned categories means that the provided explanation is extremely clear, natural, and precise, respectively. We score neutral as a 4.

Data associated with respondents that completed the survey in less than 5 minutes (half the approximate time for filling out the survey) were deleted. Furthemore, we deleted the data of participants which have selected the same response to every question, regardless of the question. After cleaning the data, we ended up having 2 different scores (per explanation, and feature), corresponding to 2 different participants. The final score of an explanation for a given feature is obtained by taking the average of the 2 different scores. Therefore, we ended up having 1 single score per explanation, and feature. Indicatively, in Table 5 we show the data collected for the explanations of the CORDIS query with id 9, and the obtained final scores. From those final scores, 18 rating sets (2 databases x 3 translation systems x 3 features) consisting of 14 elements each (1 for every query), were created.

Table 5: Final scores per feature for the explanations of the CORDIS query with id 9.

Features	*Rating*	*CORDIS QUERY ID 9*		
		Logos v.1	Logos v.2	Ground truth
Clarity	Expert A	2	6	7
	Expert B	4	6	7
	Final Score	3	6	7
Fluency	Expert A	1	3	7
	Expert B	4	4	7
	Final Score	2.5	3.5	7
Precision	Expert A	7	7	7
	Expert B	6	7	7
	Final Score	6.5	7	7

In Table 4, we show the averages of those sets, accompanied with their standard deviation between brackets. Logos v.2 leads to better translations in terms of clarity and fluency for both databases (average score increases). However, we see that these scores do not surpass those of the ground truth. An interesting observation is that as the explanations become clearer and more fluent, precision decreases. In other words, as the explanations become more natural, they tend to lose their ability to explicitly explain each part of their associated SQL query. Lastly, we see that the difference between the average fluency scores of Logos v.2 and Logos v.1, increases for the SDSS database. As mentioned in the previous Section, this is due to the nature of the SDSS database which has a less explainable database schema in terms of NL explanation. By adopting labels for the components of the database schema (tables, attributes, and joins), we increase the average fluency score (Logos v.2).

6 CONCLUDING REMARKS

In this paper, the translation of SQL queries has been discussed as a potential solution to problems that rise in data exploration. In that vein, we have extended the graph-based model of Logos [3]. The system's capabilities have been extended to two different directions: (a) that of translating different types of queries and (b) that of creating fluent explanations. To this end, we have implemented changes in the system's parser, introduced new types of nodes and edges for the query graph, created new translation rules and templates, and finally modified the available traversal strategies (BST, MRP, TMT). In addition, a data set composed of 28 queries coming from different user cases (astronomy, policy making) has been created to perform both automated and human evaluation.

The experiments highlighted the following two challenges: (a) the need for creating automated metrics which do not consider only the ground truth and (b) the need for generating different explanations for different groups of people. Regarding the latter, we observed that: (a) scientific related databases are far more difficult to be explained by natural language and (b) sometimes the scientists themselves ask for less detailed interpretations. Lastly, it has been observed that as the explanations become more natural, they tend to lose their ability to explicitly explain each part of the query. Depending on the application, one may want to have more precise or more natural explanations.

Having said all this, SQL to text is a hard problem to solve and there is no single solution. Different explanations may be suitable for different applications or even different groups of people, albeit all correct. As future work, we plan to improve our system by adding an ontology which provides mappings for the interpretation of several parts of the query graph.

ACKNOWLEDGMENTS

This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement (No 863410). We also thank the Intelligent Open Data Exploration (INODE) team for contributing on the system's evaluation.

REFERENCES

[1] Sihem Amer-Yahia, Georgia Koutrika, Frederic Bastian, Theofilos Belmpas, Martin Braschler, Ursin Brunner, Diego Calvanese, Maximilian Fabricius, Orest Gkini, Catherine Kosten, Davide Lanti, Antonis Litke, Hendrik Lücke-Tieke, Francesco Alessandro Massucci, Tarcisio Mendes de Farias, Alessandro Mosca, Francesco Multari, Nikolaos Papadakis, Dimitris Papadopoulos, Yogendra Patil, Aurélien Personnaz, Guillem Rull, Ana Claudia Sima, Ellery Smith, Dimitrios Skoutas, Srividya Subramanian, Guohui Xiao, and Kurt Stockinger. 2021. INODE: Building an End-to-End Data Exploration System in Practice [Extended Vision]. *CoRR* abs/2104.04194 (2021). arXiv:2104.04194 https://arxiv.org/abs/2104.04194

[2] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing Source Code using a Neural Attention Model. In *ACL*. The Association for Computer Linguistics.

[3] Andreas Kokkalis, Panagiotis Vagenas, Alexandros Zervakis, Alkis Simitsis, Georgia Koutrika, and Yannis Ioannidis. 2012. Logos: a system for translating queries into narratives. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. 673–676.

[4] Georgia Koutrika, Alkis Simitsis, and Yannis E Ioannidis. 2010. Explaining structured queries in natural language. In *ICDE*. IEEE, 333–344.

[5] Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak SPARQL: translating SPARQL queries into natural language. In *Proceedings of the 22nd international conference on World Wide Web*. 977–988.

[6] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[7] Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Nat. Lang. Eng.* 3, 1 (1997), 57–87. https://doi.org/10.1017/S1351324997001502

[8] Alkis Simitsis and Yannis E. Ioannidis. 2009. DBMSs Should Talk Back Too. In *CIDR*.

[9] Alkis Simitsis, Georgia Koutrika, Yannis Alexandrakis, and Yannis Ioannidis. 2008. Synthesizing structured text from logical database subsets. In *EDBT*. 428–439.

[10] Chris van der Lee, Emiel Krahmer, and Sander Wubben. 2018. Automated learning of templates for data-to-text generation: comparing rule-based, statistical and neural methods. In *Proceedings of the 11th International Conference on Natural Language Generation*. 35–45.

[11] Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2018. Learning neural templates for text generation. *arXiv preprint arXiv:1808.10122* (2018).

[12] Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. SQL-to-Text Generation with Graph-to-Sequence Model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). 931–936.

7 APPENDIX

Table 6: The 14 SQL queries of the CORDIS database.

ID	CORDIS SQL Queries
1	SELECT sum(pm.ec_contribution) AS funding_received FROM projects p, project_members pm WHERE pm.project=p.unics_id AND p.framework_program='H2020';
2	SELECT c.name FROM institutions i, countries c WHERE c.unics_id=i.country_id AND i.name='Athena';
3	SELECT pr.title FROM projects pr, project_subject_areas psa, subject_areas sa WHERE pr.unics_id = psa.project AND psa.subject_area = sa.code AND sa.title = 'Mathematics and Statistics';
4	SELECT distinct t.title FROM projects pr, project_topics pt, topics t WHERE pr.unics_id = pt.project AND pt.topic = t.code AND pr.end_year = 2014;
5	SELECT p.full_name FROM people p, projects pr, project_topics pt, topics t WHERE p.unics_id = pr.principal_investigator AND pr.unics_id = pt.project AND pt.topic = t.code AND t.title = 'Systems';
6	SELECT m.title FROM people p, projects pr, project_programmes pm, programmes m WHERE p.unics_id = pr.principal_investigator AND pr.unics_id = pm.project AND pm.programme = m.code AND p.full_name = 'Thomas Bell';
7	SELECT p.acronym FROM projects p, project_members pm, institutions i, countries c WHERE p.unics_id = pm.project AND pm.institution_id = i.unics_id AND i.country_id = c.unics_id AND c.country_name = 'Greece';
8	SELECT pe.full_name FROM projects pr, people pe WHERE pr.principal_investigator = pe.unics_id AND pr.start_year = 2014;
9	SELECT i.institutions_name FROM institutions i, countries c WHERE i.country_id = c.unics_id AND c.country_name = 'France';
10	SELECT mb.member_name FROM project_members mb, activity_types a WHERE a.code = mb.activity_type AND a.description = 'Research Organisations';
11	SELECT count(p.title) FROM projects p GROUP BY p.start_year;
12	SELECT count(i.name) FROM institutions i, countries c WHERE i.country_id=c.unics_id GROUP BY c.name;
13	SELECT title FROM topics WHERE title like '%climate%';
14	SELECT count(p.title) FROM projects p WHERE p.start_year=2018;

Table 7: The 14 SQL queries of the SDSS database.

ID	SDSS SQL Queries
1	SELECT objid FROM photoobj WHERE clean=1;
2	SELECT specobjid, z FROM specobj WHERE class = 'QSO' AND zwarning = 0;
3	SELECT objid FROM photoobj WHERE ra >185 AND ra <185.1 AND dec <5;
4	SELECT specobjid FROM specobj WHERE survey = 'segue2';
5	SELECT specobjid FROM specobj WHERE class = 'STAR' AND zwarning = 0;
6	SELECT s.specobjid FROM specobj as s WHERE s.subclass = 'STARFORMING' AND s.class= 'GALAXY';
7	SELECT s.specobjid FROM specobj as s WHERE s.subclass = 'OB' AND s.class= 'STAR';
8	SELECT * FROM photoobj WHERE ra >100 and dec <100 AND type = 3;
9	SELECT DISTINCT type FROM photoobj;
10	SELECT class, count(*) FROM specobj GROUP BY class;
11	SELECT g.* FROM specobj s, galspecline g WHERE s.specobjid = g.specobjid and ra <185 AND dec <25;
12	SELECT n.* FROM neighbors n, photoobj p WHERE p.objid = n.objid AND p.b = 1.072 and p.l = 174.535;
13	SELECT s.* FROM specobj s, galspecline g WHERE s.specobjid = g.specobjid;
14	SELECT p.u, p.g, p.r, p.i, p.z FROM specobj s, photoobj p WHERE s.bestobjid = p.objid AND s.class = 'QSO';