A Large Language Model approach to SQL-to-Text Generation

Vanessa Câmara†, Rayol Mendonca-Neto†, André Silva†, Luiz Cordovil-Jr†
†*Sidia Science and Technology Institute, Av. Darcy Vargas, 654, 69055-035, Manaus, AM, Brazil*
E-mail: {vanessa.camara, rayol.neto, andre.fernandes, luiz.cordovil}@sidia.com

2024 IEEE International Conference on Consumer Electronics (ICCE) | 979-8-3503-2413-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICCE59016.2024.10444148

***Abstract*—Generating relevant explanations given an structured code representation, such as SQL, is a challenging task. Tackling the SQL-to-text, more specifically the SQL-explanation problem, benefits both non-technical and technical users. Automatic explanations written in human language can facilitate the understanding of the query's logical structure and it also helps developers to better document and learn SQL code. The approaches for this niche are diverse. Some of them involve sequence-to-sequence models and others utilize graph-to-sequence models to generate explanations. However, considering the latest advances in Large Language Models (LLMs) and the relatively little attention in SQL-to-text problem, we investigate a new generative approach based on LLMs to infer the logical structure about the query, including columns, tables and relations. We categorize our research on SQL-explanation as a subtask of SQL-to-text to differ from the translation of SQL code into natural language questions. Experiments were conducted with the open-source Falcon LLM and compared with T5 LLM and Graph2Seq models. The results show that Falcon outperforms previous models achieving 70% of accuracy with human evaluation on Spider dataset and it achieves competitive 75% accuracy with human evaluation on WikiSQL dataset.**

***Index Terms*—Deep Learning, Natural Language Processing, SQL-to-Text, Pre-trained Models, Large Language Models**

I. Introduction

Structured Query Language (SQL) is a widely adopted tool in order to access, navigate and obtain data from a variety of data stores, such as relational databases [1]. Writing SQL code for complex queries usually requires software development skills. However, recent advances in natural language interfaces (NLI) powered by artificial intelligence (AI) and machine learning (ML) are revolutionizing the industry with automatic translation of natural language questions (NLQ) into SQL, and further leveraging the potential of big data to non-technical users. When the NLQ is received in written form, the task of translating it is called text-to-SQL.

On the other hand, there are important applications of the reverse task, i.e., the automatic translation of SQL code into human language, which is usually referred to as SQL-to-text in the literature. The SQL-to-text problem can be seen as a text summarization problem, which aims at generating natural language text from SQL code. However, it is worthy mentioning that the term SQL-to-text may refer to different tasks depending on the intended application, such as SQL-to-NLQ, SQL-explanation and table-answering.

For instance, the translation of SQL into NLQ produces an equivalent of *what is the underlying question* that the SQL query is attempting to answer. Besides code written by humans, SQL-to-NLQ has also been applied to automatic documentation of machine-generated code, which produces NLQ and SQL pairs to alleviate the annotation scarcity problem of text-to-SQL [2], [3].

SQL-explanation is a different task. It produces an automatic *explanation of the SQL code itself* providing information about the tables, columns, functions and clauses involved in the query. Documenting a SQL query's logical structure and explaining in natural language its columns and tables can be very educational for software developers, who must learn structured code. This process can shorten the learning curve, speed up programming ability [4], and reduce the technical debt of legacy code.

Yet another related task consists in executing the SQL query and generating an automatic *explanation and/or summary about the data returned by the query*. This task is referred to as table-answering. The interpretability of SQL queries can be counterintuitive for non-technical users, such as clients, that must query information for data insights, for instance. These are examples of data democratization [5], [6], which allows all users to easily access data and derive value from it no matter their technical knowledge.

There are many works about SQL-to-NLQ and table-answering in the literature. However, the SQL-explanation is still underexplored [1], [7], although there are notable works in this area. For instance, Xu et al. [8] propose a strategy to represent the SQL query as a directed graph reaching 79.2% of accuracy with human evaluation, considering interpretation on both the correctness and grammatical cohesion conforming to the SQL input. Katsogiannis-Meimarakis et al. [1] proposed a Transformer-based [9] sequence-to-sequence model following an encoder-decoder architecture and reported 58.57% accuracy on the test dataset with human evaluation.

In this context, Large Language Models (LLMs) are the state of the art at generative summarization and other NLP tasks. There are many pre-trained LLMs in the literature such as BERT [10], T5 [11], LLaMa [12], Falcon [13] and GPT [14]. Despite that, the potential of these models have not been fully explored for text-to-SQL and SQL-to-text tasks [1], [15], [16]. Considering all those challenges, it urges for more

This paper is a result of the Research, Development & Innovation Project performed at Sidia Science and Technology Institute sponsored by Samsung Eletrônica da Amazônia Ltda., using resources under terms of Brazilian Informatics Law No. 8.387/1991, by having its disclosure and publicity in accordance with art. 39 of Decree No. 10.521/2020.

research in the area in order to improve data democratization.

In this work, we assess the performance of a LLM model at the SQL-explanation task. Experiments were conduced with the open-source Falcon-7B-Instruct Pre-trained Language Model without fine-tuning, and its performance was compared to T5 LLM and Graph2Seq models. Results show that the model achieves impressive 70% accuracy with human evaluation on Spider dataset [17], which outperforms previous models, and achieves competitive 75% accuracy with human evaluation on WikiSQL [18] dataset.

The paper is organized as follows. Section II presents the related works about SQL-explanation. Section III describes the methodology, datasets and models for the experiments. Section IV presents the results. Section V is the conclusion.

II. Related Work

The work of Xu et al. [8] proposes a strategy to represent the SQL query as a directed graph and then employ a graph-to-sequence model to encode the global structure information into node embeddings, reaching 79.2% of accuracy with human evaluation on WikiSQL dataset [3], considering interpretation on both the correctness and grammatical cohesion conforming to the SQL input. However, it was only tested on queries taken from the WikiSQL dataset, which comprises SQL queries only for single tables without joins, and thus it has relatively low complexity.

Katsogiannis-Meimarakis et al. [1] proposes the use o the T5-base Pre-trained Language Model, which is a Transformer-based model [9] that performs sequence-to-sequence following an encoder-decoder architecture. The results on human evaluation, whose experts were asked to classify the prediction, are 58.57% correct on Spider dataset [17]. However, even though deep learning solutions offer better generalization to unseen databases and more fluent explanations, they are not guaranteed to generate accurate explanations every time. Furthermore, in therms of LLMs, the T5 model is relatively small with 220 million parameters, which might not be sufficient for the SQL-explanation task. Low performance in the NLP tasks comes from the fact that performance scales up with model size [19].

III. Methodology

A. Dataset

In our experiments we employ the Spider and WikiSQL datasets. Spider is a large-scale, complex and cross-domain semantic parsing and text-to-SQL dataset annotated by 11 college students. It consists of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables, covering 138 different domains. The authors define a complex and cross-domain semantic parsing and a text-to-SQL task since it uses multiple databases and different schemata in the train set and the test set.

The WikiSQL is a collection of 80.654 hand-crafted question-SQL pairs along with 24,241 HTML tables collected from Wikipedia. The tables are collected from Bhagavatula et al. [20], and the small tables that have less than five columns or five rows are filtered.

B. Falcon-7B-Instruction

The Falcon Large Language Model[1] is a causal decoder-only model fine-tuned on 1,500 billions tokens on RefinedWeb [21] instruct dataset. It has seven billions parameters and it is open-source. It features an architecture optimized for inference, with FlashAttention [22] and multiquery [23] and is available under the Apache 2.0 license. Falcon-7B-Instruct is an instruct model, which means that this language model is provided with explicit instructions or guidance during the text generation process. This could involve providing specific instructions or constraints to influence the output of the model. We experiment with this model in a zero-shot setting, that is, the model predicts query explanations given unseen SQL queries without fine-tuning it for specific SQL-to-text datasets. The Falcon-7B model is a strong base model that outperforms comparable open-source models, such as MPT-7B[2], StableLM[3], and RedPajama[4] available at HuggingFace.

C. Experiment Description

The diagram on Fig. 1 shows how to generate SQL-explanations with a pre-trained LLM. We prompt the following natural language instruction: “Given a SQL query, tell me in natural language what it returns and describe each column. SQL query: ” followed by the corresponding SQL code. Then, the model will produce an explanation of the code.

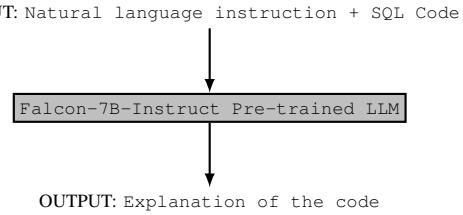

Fig. 1. Generating SQL-explanations with pre-trained LLM.

D. Accuracy with Human Evaluation Metric

For the accuracy with human evaluation metric, we randomly sampled SQL queries to be explained by the model. We asked two software developers to rate each interpretation against the correctness conforming to the input SQL. For WikiSQL dataset, we evaluated 100 random queries and for Spider dataset we evaluated 350 random queries.

IV. Results

Table I summarizes the results of our experiments with the Falcon model compared to T5 results reported by Katsogiannis-Meimarakis et al. [1] and Graph2Seq results reported by Xu et al. [8] on Spider and WikiSQL datasets,

[1]https://huggingface.co/tiiuae/falcon-7b
[2]https://huggingface.co/mosaicml/mpt-7b
[3]https://huggingface.co/stabilityai/stablelm-base-alpha-7b-v2
[4]https://huggingface.co/togethercomputer/RedPajama-INCITE-7B-Base

respectively. We can see that Falcon achieved 70.28% accuracy with human evaluation on Spider dataset, which is significantly better than the 58.58% accuracy achieved by T5. One possible explanation is that the performance of the LLM model scales better with larger models, since Falcon has billions of parameters compared to T5 with millions. On WikiSQL, Falcon achieved 75.00% accuracy, which is similar performance when compared to Graph2Seq-NGE, however generating inferior explanations when compared to Graph2Seq-PGE. Furthermore, our accuracy estimation for Falcon on WikiSQL was based on a sample with a 100 random queries, which is a smaller sample when compared to the experiments reported for the Graph2Seq models.

TABLE I
HUMAN EVALUATION OF MODELS ON SQL-EXPLANATION

Model	Dataset	Human Eval.	Sample Size
Falcon-7B-Instruct	Spider	**70.28%**	350
T5-base	Spider	58.57%	350
Graph2Seq-PGE	WikiSQL	**79.20%**	1000
Graph2Seq-NGE	WikiSQL	75.30%	1000
Falcon-7B-Instruct	WikiSQL	75.00%	100

Table II shows eight examples of SQL-explanations generated automatically by the Falcon model for SQL queries with different complexities on Spider dataset. The NLQ and the corresponding SQL query are part of the Spider dataset, but only SQL query is used as input to the Falcon model with the instruction to generate the corresponding explanation. Hence, the NLQ question column is provided only for better understanding of the SQL query as well as better assessment of the SQL-explanation generated by the model. The first five examples show good explanations generated by the model, which could handle WHERE clauses, ORDER BY clauses, subqueries, GROUP BY clauses, and joins. The last three examples show bad explanations with mistakes such as foreing key misclassification, mistaking a table for a column, and failure to referring to object names properly, which could be easily inferred from the SQL code.

V. CONCLUSION

In this work, we reported experiments with Falcon and compared it with T5 and Graph2Seq models. Results indicate that pre-trained open-source LLMs achieve good performance for SQL-explanation. However, there is plenty of room for improvement, since these models still make serious mistakes producing wrong explanations concerning database schemata.

In future works, we would like to experiment with other open-source pre-trained LLMs such as LLaMA in both zero-shot setting and fine-tuning. Also, it would be interesting to compare their performance with ChatGPT. Finally, we would like to investigate mechanisms to improve SQL-explanations about the database schemata implicit in the SQL code.

REFERENCES

[1] G. G. Katsogiannis-Meimarakis, "Data democratisation with deep learning: Structured query translation from and to natural language," 2023.

[2] X. Xu, C. Liu, and D. Song, "Sqlnet: Generating structured queries from natural language without reinforcement learning," *arXiv preprint arXiv:1711.04436*, 2017.

[3] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[4] A. Simitsis and Y. Ioannidis, "Dbmss should talk back too," *arXiv preprint arXiv:0909.1786*, 2009.

[5] G. Katsogiannis-Meimarakis and G. Koutrika, "A survey on deep learning approaches for text-to-sql," *The VLDB Journal*, pp. 1–32, 2023.

[6] J. Sen, C. Lei, A. Quamar, F. Özcan, V. Efthymiou, A. Dalmia, G. Stager, A. Mittal, D. Saha, and K. Sankaranarayanan, "Athena++ natural language querying for complex nested sql queries," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2747–2759, 2020.

[7] N. Deng, Y. Chen, and Y. Zhang, "Recent advances in text-to-sql: a survey of what we have and what we expect," *arXiv preprint arXiv:2208.10099*, 2022.

[8] K. Xu, L. Wu, Z. Wang, Y. Feng, and V. Sheinin, "Sql-to-text generation with graph-to-sequence model," *arXiv preprint arXiv:1809.05255*, 2018.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[11] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[12] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[13] Y. X. ZXhang, Y. M. Haxo, and Y. X. Mat, "Falcon llm: A new frontier in natural language processing," *AC Investment Research Journal*, vol. 220, no. 44, 2023.

[14] C. Ai, "Gpt-4 by admin march 27th, 2023 no comments 19 min read gpt-4 technical report."

[15] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, E. Goffinet, D. Heslow, J. Launay, Q. Malartic *et al.*, "Falcon-40b: an open large language model with state-of-the-art performance," Technical report, Technology Innovation Institute, Tech. Rep., 2023.

[16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[17] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," *arXiv preprint arXiv:1809.08887*, 2018.

[18] W. Hwang, J. Yim, S. Park, and M. Seo, "A comprehensive exploration on wikisql with table-aware word contextualization," *arXiv preprint arXiv:1902.01069*, 2019.

[19] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[20] C. S. Bhagavatula, T. Noraset, and D. Downey, "Methods for exploring and mining tables on wikipedia," in *Proceedings of the ACM SIGKDD workshop on interactive data exploration and analytics*, 2013, pp. 18–26.

[21] G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei, and J. Launay, "The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only," *arXiv preprint arXiv:2306.01116*, 2023.

[22] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 344–16 359, 2022.

[23] N. Shazeer, "Fast transformer decoding: One write-head is all you need," *arXiv preprint arXiv:1911.02150*, 2019.

TABLE II
EXAMPLES OF SQL-EXPLANATION GENERATED BY THE FALCON MODEL ON THE SPIDER DATASET

NLQ	**SQL**	**SQL-explanation**
What are the names of the heads who are born outside the California state?	`SELECT name FROM head WHERE born_state != 'California'`	The query returns a list of names that are not from California. The name column contains information about each individual, while the WHERE clause filters out California-born individuals.
List the titles of the books in ascending order of issues.	`SELECT Title FROM book ORDER BY Issues ASC`	The query returns a list of book titles in ascending order based on the 'Issues' column, which likely indicates the amount or type of issue(s) present in each book.
What are the ids of all employees that don't have certificates?	`SELECT eid FROM Employee EXCEPT SELECT eid FROM Certificate`	The query returns a list of unique employee IDs, excluding those found in a separate list of certificate IDs. The 'eid' column in both statements refers to 'Employee ID'.
In which year were most departments established?	`SELECT creation FROM department GROUP BY creation ORDER BY count(*) DESC LIMIT 1`	This query selects the 'creation' column, which likely stores the names or dates of when a department was established. The 'GROUP BY' clause groups the results by the 'creation' column, and the 'COUNT(*)' function counts the number of rows in each group. The 'DESC' keyword sorts the results in descending order based on the 'count(*)' column. The 'LIMIT 1' clause ensures only one row is returned.
What are the ids of all students for courses and what are the names of those courses?	`SELECT T1.student_id , T2.course_name FROM student_course_registrations AS T1 JOIN courses AS T2 ON T1.course_id = T2.course_id`	The given SQL query returns a list of student IDs and the names of the courses they're enrolled in. The first column, 'student_id', is an identifier for each student in the table. The second column, 'course_name', is the name of the course they're currently enrolled in. This is the column where we find the list of courses that each student is currently enrolled in. Together, they form a relationship to show the student's course history.
Show the years and the official names of the host cities of competitions.	`SELECT T2.Year, T1.Official_Name FROM city AS T1 JOIN farm_competition AS T2 ON T1.City_ID = T2.Host_city_ID`	The select statement returns a table containing two columns named 'T2.Year' and 'T1.Official_Name'. The 'T1.Official_Name' column lists the name of each city that is being compared, while the 'T2.Year' column indicates the year in which each city participated in the farm competition. *The 'city_ID' column is a foreign key that references the 'Host_city_ID' column in the 'farm_competition' table*, creating a relationship between the two tables.
What are the students ids of students who have more than one allergy?	`SELECT StuID FROM Student EXCEPT SELECT StuID FROM Has_allergy`	The query returns the unique student IDs that do not have the condition 'allergy'. The first column (StuID) is a primary key column in the Student table and is used to identify each unique student in the table. *The second column (Has_allergy) is an example of a column name in the query*, which filters the results to only include student IDs that do not have the condition 'allergy'.
List the creation year, name and budget of each department.	`SELECT creation, name, budget_in_billions FROM department`	The query returns a list of columns with corresponding names: 'creation' refers to when *an object* was created, 'name' is the name of *the object*, and 'budget_in_billions' represents the total budget for *that object* in billions.