2024 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) | 979-8-3503-6613-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/VL/HCC60511.2024.00048

Understanding the Performance of Large Language Model to Generate SQL Queries

Minhyuk Ko, Dibyendu Brinto Bose, Weilu Wang, Mohammed Seyam, Chris Brown
Department of Computer Science
Virginia Tech, Blacksburg, VA, USA
{minhyukko, brintodibyendu, weilu, seyam, dcbrown}@vt.edu

***Abstract*—Recent developments in Artificial Intelligence (AI) have shifted the software development paradigm. Past studies demonstrated how effective AI can generate code for programming purposes. However, to our knowledge, no prior study has been done to evaluate the effectiveness of SQL queries generated by AI. We utilized nine AI assistants to generate SQL queries. Our results reveal that most AI assistants generate inaccurate SQL queries, and based on the results, we provide possible implications for SQL developers.**

***Index Terms*—AI Programming Assistants, Usability Study, Database Systems**

I. Introduction

Structured Query Language (SQL) is one of the most popular languages for managing relational databases [30] and one of the most used languages across programming in general [32]. However, the complexity of queries and the declarative nature of SQL can make SQL programming more difficult to learn compared to procedural or object-oriented programming [10]—in particular for novice and non-expert programmers [28], [31].

To support end-user programmers handling of data in database management systems (DBMS), various *text-to-SQL* systems have been introduced to bridge the gap between users and data by translating natural language into queries for relational databases [14]—nowadays often leveraging machine learning (ML) and natural language processing (NLP) techniques [16]. For example, NaLIR is a tool that can translate natural language, like English, to SQL queries [21].

However, current text-to-SQL systems face challenges limiting their adoption in practice. Prior work suggests state-of-the-art approaches are inaccurate, making mistakes, especially for more complex queries [31]. Further, text-to-SQL systems can face issues with usability, inhibiting their adoption. Recently, the rapid emergence of LLMs has changed and transformed various software engineering processes, including code generation [34], code summarization [3], translating code between different programming languages [26], and conversing with programmers to complete software development tasks [27].

Our project extends this work by investigating the capabilities of commercial LLM-based systems for producing SQL queries when an English statement is provided. We conduct a comparative analysis to investigate SQL query generation across a wide range of off-the-shelf LLM-based programming assistants and AI chatbots—including ChatGPT, GitHub CoPilot, Amazon Q Developer, Codeium, Blackbox, CodeGeeX, Tabnine, Bing Chat, and Google Gemini. They were chosen as they are publicly available assistants without any modifications such as fine-tuning. Our research will open up discussions on how LLM-based systems can be used to generate SQL queries and extend various works [17], [22] that aim to enhance developers' productivity.

II. Data Collection and Evaluation

To investigate the text-to-SQL performance of off-the-shelf LLM-based assistants, we first analyze the capabilities of these systems to generate accurate SQL queries based on natural language prompts.

A. Tool Sampling.

We used various AI programming assistants and LLM-based chatbots to observe their capabilities to support database development. In this paper, we define *AI programming assistants* as tools that are primarily designed to support assistance in programming, such as GitHub Copilot [11], Blackbox [1], Tabnine [29], Q Developer [2], and Codeium [7]. We define *Large Language Models (LLMs)* as chatbots that are not specifically intended for programming assistance, such as ChatGPT [25], Google Gemini [12], and Bing Chat [4]. We define *AI assistants* as all tools that can assist in writing code through AI, including AI programming assistants and Large Language Models.

B. Study Database.

We used the European Soccer Database [24], which is one of the most popular repositories on Kaggle [15], for our evaluation. The dataset has information regarding more than 25,000 matches and 10,000 players from 11 European Countries with their lead championship between seasons 2008 and 2016. The attributes of players and teams are sources from EA Sports' FIFA video game series. The information is composed in the dataset of seven tables: Country, League, Match, Player, Player_Attributes, Team, Team_Attributes. Each table contains several rows ranging from 11 to 183978 and several columns ranging from 2 to 115.

979-8-3503-6613-6/24/$31.00 ©2024 IEEE
DOI 10.1109/VL/HCC60511.2024.00048

TABLE I: Accuracy Results for each LLM

AI Tool/LLM	Accuracy (% of test cases passed)
ChatGPT	55.0% ($n = 39$)
Amazon Q Developer	50.7% ($n = 36$)
Tabnine	21.1% ($n = 15$)
Blackbox	16.9% ($n = 12$)
Bing Chat	14.1% ($n = 10$)
Google Gemini	11.3% ($n = 8$)
GitHub Copilot	0% ($n = 0$)
Codeium	0% ($n = 0$)
CodeGeeX	0% ($n = 0$)
Total:	18.78% ($n = 120/639$)

C. Query Prompt.

Prompting is a key feature of LLMs, where users input natural language and receive automatically generated responses in a natural language format [35]. Here, we provided a subset of the prompts that we used for SQL query generation. The complete prompts are provided in the supplemental materials [19].

- Display the id, birthday, height, weight who is a player.
- List all the teams that have won more than half of the matches that they played
- Adam Brown has gained 20 pounds due to eating too many hamburgers. Update the player table accordingly.

D. Accuracy Measurement

To test out the queries that were generated by AI assistants, we utilized ChatGPT [13] to create unit testing script in Python. Then, we executed the script. Based on how many test cases the query can pass, we would be able to measure its accuracy in retrieving the information.

E. Quality Measurement

To evaluate the quality of the SQL queries, we measured indentation [20], naming convention, number of comments present in the query [6], whether the query has an error handling mechanism [5], and whether the query is easily be updated [23]. We developed a web application [18] where, after providing SQL queries, it will return the readability report with a correctly indented version of the code and also an error mechanism policy if possible.

F. Preliminary Findings

We found most AI assistants are unable to generate accurate SQL queries using natural language prompts. The average accuracy across the AI tools is less than 19%. Three AI assistants (GitHub Copilot, Codeium, and CodeGeeX) were unable to generate any accurate SQL queries based on our evaluation test cases. The most accurate LLM was ChatGPT where, out of 71 test cases, 39 passed and 32 failed. These results are presented in Table II, where we presented the accuracy results for each AI tools. Further manual analysis provided several insights into why LLMs were inaccurate.

a) Missing Columns: In some instances, in particular test failures for ChatGPT in SQL query generation, inaccuracies were due to missing columns in the generated table. We traced back to the schema and found out that the generated table did not follow the instructions mentioned in the prompts.

b) Incomplete Tests: Another notable finding is that, in some cases, test scripts were incomplete. For instance, we observed ChatGPT provided frameworks with "..." in functions instead of complete code. In future work, we should consider testing queries with man-made scripts since we evaluate the accuracy of queries or feed more explicit prompts to avoid misunderstanding.

c) Miscellaneous: We observed several other issues specific to certain AI tools that lack a clear explanation. First, Codeium [7]—despite its ability to support multiple programming languages and development tasks, [8] would always generate a one-line SQL query, regardless of the given prompt. When we provided more casual prompts, GitHub Copilot was not able to understand the prompt that we provided—causing an error in the output.

III. Proposed Research

Throughout this research, we were able to identify AI tools that were able to generate quality SQL queries. Based on our preliminary findings, we propose a user study to understand how AI tools can be utilized in SQL programming. Prior literature suggests that general programmers tend to use AI tools to find a good starting point and that they have a hard time understanding and debugging the code generated by AI tools [33]. We would like to find out if this finding holds in SQL programming.

IV. Relevance

Our work encompasses the theme of *thinking more deeply about code* and *future of work with AI* as it tries to understand how AI can be used in a particular coding situation. Two papers that discuss how AI-generated code can be utilized have been published in VL/HCC 2023 [9], [34]. Similarly, our paper discusses how AI-generated code can be used among SQL programmers. Furthermore, the special emphasis of VL/HCC 2024 is VL/HCC and Generative AI. Various preliminary findings from our study would be able to motivate various future studies on how generative AI can be used in a specific programming context.

V. Presentation

To provide a visual understanding of how AI can be utilized in SQL programming, we plan to create a poster. We also plan to bring a laptop to demonstrate our web application [18] used to evaluate the quality of queries. We believe this will foster a two-way conversation between the presenter and the audience.

APPENDIX

This information is based on the period of the study (November 2023).

TABLE II: LLM Information

AI Tool/LLM	GPT Version	Number of Tokens	Number of Parameters
ChatGPT	3.5	4096	17B
Amazon Q Developer	Unknown	Unknown	Unknown
Tabnine	3.5	Unknown	2B
Blackbox	LLaMA	256K	1.5B
Bing Chat	4.0	25K	175B
Google Gemini	N/A	128K	137B
GitHub Copilot	3.0	4096	12B
Codeium	Unknown	4096	Unknown
CodeGeeX	2.0	158B	13B

REFERENCES

[1] Blackbox. https://www.blackbox.ai/. Accessed: June 19, 2024.

[2] What is codewhisperer? https://docs.aws.amazon.com/codewhisperer/latest/userguide/what-is-cwspr.html. Accessed: June 19, 2024.

[3] Toufique Ahmed and Premkumar Devanbu. Few-shot training llms for project-specific code-summarization. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–5, 2022.

[4] Bing Chat. https://www.bing.com/chat.

[5] Miguel Cebollero, Jay Natarajan, Michael Coles, Miguel Cebollero, Jay Natarajan, and Michael Coles. Error handling and dynamic sql. *Pro T-SQL Programmer's Guide*, pages 589–612, 2015.

[6] Joe Celko. *Joe Celko's SQL programming style*. Elsevier, 2005.

[7] Codeium. https://codeium.com/.

[8] Yonas Enchalew. Five alternatives to github copilot, 2021.

[9] Kasra Ferdowsi, Jack Williams, Ian Drosos, Andrew D Gordon, Carina Negreanu, Nadia Polikarpova, Advait Sarkar, and Benjamin Zorn. Coldeco: An end user spreadsheet inspection tool for ai-generated code. In *2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 82–91. IEEE, 2023.

[10] Philip Garner and John Mariani. Learning sql in steps. *Learning*, 12:23, 2015.

[11] GitHub Copilot. https://github.com/features/copilot.

[12] Google. Google bard. https://www.google.com/bard.

[13] Vitor Guilherme and Auri Vincenzi. An initial investigation of chatgpt unit test generation capability. In *Proceedings of the 8th Brazilian Symposium on Systematic and Automated Software Testing*, pages 15–24, 2023.

[14] Gary G. Hendrix, Earl D. Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum. Developing a natural language interface to complex data. *ACM Trans. Database Syst.*, 3(2):105–147, jun 1978.

[15] Kaggle. https://www.kaggle.com/.

[16] George Katsogiannis-Meimarakis and Georgia Koutrika. A survey on deep learning approaches for text-to-sql. *The VLDB Journal*, 2023.

[17] Shawal Khalid and Chris Brown. Software engineering approaches adopted by blockchain developers. In *2023 Tenth International Conference on Software Defined Systems (SDS)*, pages 1–6. IEEE, 2023.

[18] Minhyuk Ko, Dibyendu Brinto Bose, Weilu Wang, Mohammed Seyam, and Chris Brown. Sql redability. https://github.com/brintodibyendu/SQL_redability, 2024.

[19] Minhyuk Ko, Dibyendu Brinto Bose, Weilu Wang, Mohammed Seyam, and Chris Brown. Supplemental materials for understanding the performance of large language model to generate sql queries. https://figshare.com/s/7e745548335eb1f1095f, 2024.

[20] Aristotelis Leventidis, Jiahui Zhang, Cody Dunne, Wolfgang Gatterbauer, HV Jagadish, and Mirek Riedewald. Queryvis: Logic-based diagrams help users understand complicated sql queries faster. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 2303–2318, 2020.

[21] Fei Li and Hosagrahar V Jagadish. Nalir: an interactive natural language interface for querying relational databases. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 709–712, 2014.

[22] Jenny T Liang, Maryam Arab, Minhyuk Ko, Amy J Ko, and Thomas D LaToza. A qualitative study on the implementation design decisions of developers. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 435–447. IEEE, 2023.

[23] Yoshifumi Masunaga, Yugo Nagata, and Tatsuo Ishii. Making join views updatable on relational database systems in theory and in practice. In *Proceedings of the 13th International Conference on Ubiquitous Information Management and Communication (IMCOM) 2019 13*, pages 823–840. Springer, 2019.

[24] Hugo Mathien. European soccer database, 2016.

[25] Open AI. https://openai.com/blog/chatgpt.

[26] Rangeet Pan, Ali Reza Ibrahimzada, Rahul Krishna, Divya Sankar, Lambert Pouguem Wassi, Michele Merler, Boris Sobolev, Raju Pavuluri, Saurabh Sinha, and Reyhaneh Jabbarvand. Understanding the effectiveness of large language models in code translation. *arXiv preprint arXiv:2308.03109*, 2023.

[27] Steven I. Ross, Fernando Martinez, Stephanie Houde, Michael Muller, and Justin D. Weisz. The programmer's assistant: Conversational interaction with a large language model for software development. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, IUI '23, page 491–514, New York, NY, USA, 2023. Association for Computing Machinery.

[28] Amir Shareghi Najar, Antonija Mitrovic, and Kourosh Neshatian. Eye tracking and studying examples: how novices and advanced learners study sql examples. *Journal of computing and information technology*, 23(2):171–190, 2015.

[29] Tabnine. https://www.tabnine.com/.

[30] Konstantinos Tatsis. A quantitative study on the popularity and performance of sql and nosql dbms., 2022.

[31] Yuan Tian, Toby Jia-Jun Li, Jonathan K Kummerfeld, and Tianyi Zhang. Interactive text-to-sql generation via editable step-by-step explanations. *arXiv preprint arXiv:2305.07372*, 2023.

[32] Lionel Sujay Vailshery. Most widely utilized programming languages among developers worldwide 2023. *Statista*, 2024. https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/.

[33] Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*, pages 1–7, 2022.

[34] Tianjia Wang, Daniel Vargas Díaz, Chris Brown, and Yan Chen. Exploring the role of ai assistants in computer science education: Methods, implications, and instructor perspectives. In *2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 92–102. IEEE, 2023.

[35] JD Zamfirescu-Pereira, Richmond Y Wong, Bjoern Hartmann, and Qian Yang. Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–21, 2023.