Evaluating the Text-to-SQL Capabilities of Large Language Models on Cantus Database

Anonymous ACL submission

Abstract

This study evaluates the performance of three large language models—GPT, Claude, and Grok—in generating SQL queries for Cantus Database, a specialized collection of Latin chant metadata. The models' capabilities were assessed through inference alone, without fine-tuning, using two schema configurations: one with predefined options and one without. Across 45 ground truth samples, GPT exhibited the highest performance, correctly generating 32 out of 45 queries (ignoring order) and achieving an F1 score of 0.88 when schema options were provided. It was also found that the inclusion of select schema values alongside natural language queries was found to significantly enhance the performance of all models. This research presents a novel approach to domain-specific Text-to-SQL tasks, demonstrating that high performance can be achieved without the computational overhead of fine-tuning.

1 Introduction

Text-to-SQL is a critical area of research focused on the automatic translation of natural language questions into SQL queries. It enables applications such as AI-driven database services and question-answering functionality, making it easier for non-experts to interact with database systems. In this project[1], we evaluate the capabilities of large language models (LLMs) on a novel domain: Cantus Database, a rich collection of metadata from Latin chants, accessible at `https://cantusdatabase.org`/.

1.1 Cantus Database

Cantus Database, or CantusDB (Cantus Team, 2024), is a website and database of Latin ecclesiastical chants from manuscripts and other sources around the world. It is actively maintained by developers at McGill University and is used by more than 250 academics and researchers. In Cantus Database, there are three main objects that are searchable and filterable on the site: *Chants*, *Sources*, and *Feasts*. *Chants* are individual pieces of liturgical music that include musical notation, text, and associated metadata. *Sources* refer to manuscripts or early printed books that preserve these chants. *Feasts* are specific liturgical celebrations within the Christian calendar to which chants are assigned and performed.

Cantus Database provides researchers with an effective way to locate specific data, but its search and filtering options are limited to predefined criteria, restricting users. To address this, we propose a solution that uses natural language queries, allowing users to retrieve results without relying on existing filters, offering greater flexibility in accessing information.

1.2 Large Language Models

Recent advancements in LLMs have significantly improved the Text-to-SQL task, which translates natural language queries into SQL code. This task is historically challenging due to the complexities of natural language understanding and database schema reasoning, but innovations in LLM architectures and pre-training now enable more efficient and accessible database interactions.

Among the LLMs currently available, three were selected for this research: OpenAI's ChatGPT-o1 (GPT) (OpenAI, 2024), chosen for its advanced reasoning capabilities; Anthropic's Claude 3.5 Haiku (Claude) (Anthropic, 2023), noted for its efficiency and extensive token limit; and xAI's Grok-2 (Grok) (xAI, 2024), valued for its integration of real-time web knowledge, which is particularly useful for addressing dynamic and less-documented aspects of Cantus Database.

Incorporating LLMs into website search and filtering poses challenges, primarily the high compu-

[1] The code for this project is accessible at: https://github.com/zhannaklimanova/nl-to-sql

tational costs of training new models. Additionally, changes to database structures often necessitate time-consuming updates to datasets due to the need for fine-tuning or retraining. Considering the added challenge posed by the lack of a large annotated dataset for Cantus Database, this research investigates the ability of modern pre-trained LLMs to tackle complex Text-to-SQL tasks on an unfamiliar database schema without relying on explicit fine-tuning or task-specific training.

By relying solely on inference, this work explores how effectively these models can generate SQL queries using only a natural language query, a database schema, and selected schema content. This approach reduces computational costs and offers a more efficient, sustainable solution.

2 Related Work

Generating SQL queries from natural language is a widely studied task in natural language processing. Previous research demonstrates that scaling training data and model size in generative models enables advanced functionalities, such as few-shot learning, without task-specific fine-tuning (Rajkumar et al., 2022). Notably, the work shows that models like Codex and GPT-3 perform competitively as Text-to-SQL solutions without additional fine-tuning. The study also explores the impact of varied prompts and assesses the performance of multiple models on established datasets, such as Spider, to determine cross-domain Text-to-SQL capabilities.

Other work investigates prompt engineering strategies for LLMs in Text-to-SQL tasks (Gao et al., 2023). This research systematically evaluates techniques such as question representation, in-context learning, and supervised fine-tuning on benchmark datasets like Spider and Spider-Realistic. They propose a novel solution, DAIL-SQL, which leverages these strategies to enhance Text-to-SQL performance. The authors also provide an empirical analysis of zero-shot scenarios, example selection, and organization strategies in few-shot learning, highlighting the potential of open-source LLMs for both in-context learning and fine-tuning.

Building on the widespread use of the Spider dataset in Text-to-SQL research, a recent study further investigated strategies for leveraging LLMs for this task (Roberson et al., 2024). Two main approaches were examined: fine-tuning open-source models and using closed-source models with advanced techniques. They identified seven common error types in generated queries, such as incorrect column selection, grouping errors, and faulty JOIN clauses. While closed-source models excelled in few-shot in-context learning, the authors noted that some queries flagged as incorrect still produce correct outputs, highlighting the need for improved evaluation metrics in Text-to-SQL research.

While these works provide insights into the capabilities of LLMs for Text-to-SQL tasks, they primarily focus on established datasets like Spider, which are larger and may have been exposed to the evaluated LLMs during training, potentially introducing overfitting. In contrast, in this study we employ a custom-made, domain-specific dataset derived from Cantus Database. Our research avoids fine-tuning or supervised learning by exclusively using prompt engineering to evaluate pre-trained LLMs. In addition, our work tested several prompt engineering strategies before selecting and standardizing those used in the JSON data schema. By evaluating multiple LLMs and optimizing prompt strategies, this work demonstrates the adaptability of LLMs to specialized datasets and highlights prompt engineering as an effective, lightweight approach for domain-specific Text-to-SQL tasks.

3 Methodology

Our methodology is summarized in the workflow illustrated in *Figure 1*.

3.1 Overview

Ground truth data is collected from Cantus Database using a locally deployed version of the website[2]. To capture SQL queries made on the database, a custom middleware is integrated into the codebase[3]. Additionally, a PostgreSQL database dump is used to extract the database schema. To fit within the context token limits of modern LLMs, the schema is reduced in size and complexity. The details of the reduction process are provided in Section 3.4.

Ground truth and predicted data are stored in JSON files, organized by LLM (GPT, Claude, Grok) and object type (*Chants*, *Feasts*, and *Sources*). Each entry includes the gold SQL query, output path, and natural language input. LLMs are

[2] Setup instructions are available at https://github.com/DDMAL/CantusDB/wiki.

[3] Middleware setup instructions are detailed in the project's README.md file.

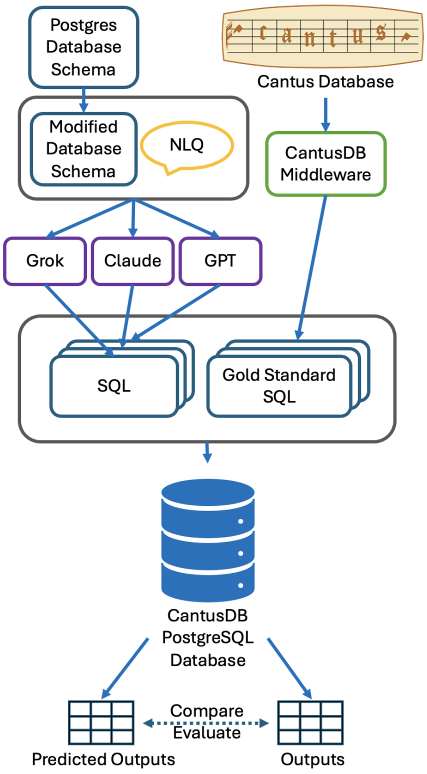

Figure 1: Overview of the methodology: Ground truth SQL queries from Cantus Database and natural language queries with a database schema are used to prompt LLMs. The generated SQL queries are executed on the PostgreSQL Cantus Database, and the resulting object IDs are compared and evaluated.

prompted with the database schema and queries to generate SQL, which is executed in a PostgreSQL Docker container to produce CSV outputs. Paths to these outputs are recorded alongside the ground truth.

3.2 Dataset

The dataset consists of 45 ground truth samples, with 15 samples for each object type (*Chants*, *Feasts*, and *Sources*). Each of the three LLMs generates predictions for 15 natural language input samples per object type, producing a total of 45 predicted SQL queries and corresponding outputs per schema. For both schema variations (with and without options) and all three object types, this totals 135 predicted outputs per schema and 270 combined. This structured approach enables a thorough evaluation of the LLMs' performance across different schema complexities and object types.

3.3 Prompting Large Language Models

To generate SQL queries using the LLMs, we create prompts that integrate the modified database schema with natural language inputs. To clearly differentiate values and attributes from regular text, we enclose values in single quotes and capitalize attribute names. An example of such a prompt is:

> *"Given this database schema, generate a SQL query that shows me all the Sources in the 'CANTUS Database' Segment as a Complete Source, from the Country 'Germany' and Provenance 'Aachen', sorted by institution siglum and source shelfmark. Format your response without any formatting or newlines."*

Once the natural language inputs and schemas are finalized, they are provided as prompts to the GPT, Claude, and Grok models. Each LLM generates an SQL query in response to the given prompts. These queries are executed on the PostgreSQL database on CantusDB, producing outputs that include lists of IDs corresponding to the search results for *Chants*, *Sources*, or *Feasts*—objects aligned with the search pages on Cantus Database website. The generated outputs are compared to the ground truth results, i.e., the SQL query outputs obtained directly from the PostgreSQL database. The evaluation methodology, detailed in Section 3.5, combines various techniques to assess the accuracy and usability of the SQL queries generated by the LLMs within the context of Cantus Database.

3.4 Prompt Engineering

The LLMs are provided with a modified database schema and a natural language input as context. The schema is generated by systematically processing the PostgreSQL database dump through the following steps:

1. Extract the database schema using PostgreSQL's schema-only dump feature.

2. Filter out only the table creation statements from the schema dump.

3. Include essential constraints, such as primary keys, foreign keys, and references.

4. Simplify the schema by removing non-essential modifiers, such as deferrable constraints.

5. Clean up leading whitespace and other formatting inconsistencies.

6. Integrate the fixed values into the database schema to produce the extended version.

In the sixth step, we introduce an extended version (with options) of the database schema that includes fixed values relevant for filtering data on Cantus Database website but absent in the raw schema. This extended schema aims to evaluate whether adding such contextual details improves the performance of SQL queries generated by the LLMs. The fixed values are outlined below.

- External identifiers for standard metadata repositories: RISM Online, VIAF, Wikidata, GND, Bibliothèque Nationale de France, Library of Congress, and DIAMM.

- Source completeness statuses: Full Source, Fragment, Reconstruction, and Fragmented.

- Temporale and Sanctorale prefixes used for Feast filtering.

- Proofreading status: Any, Yes, or No.

3.5 Evaluation Metrics

Five quantitative evaluation metrics were used to gauge the performance of the different LLMs. The first verified if the SQL queries generated by the LLMs returned the correct elements, ignoring the order. The second metric was the ordered metric, which made sure the items were in the correct order. This was done since Cantus Database search requires the user to specify an order of the items, and this order was reflected in the natural language query. These two metrics were treated as raw counts, so a value of 32 means that 32 out of the 45 queries were correct. Since there are 45 queries in total, both metrics are out of 45.

Precision, recall, and F1 were used to assess the performance. Precision measures how many items the LLM generates are correct, reflecting how well the model avoids generating irrelevant items. Whereas recall evaluates how many correct items from the gold standard were generated by the model, indicating its ability to capture all relevant results. To balance precision and recall, the F1 score was also calculated. The F1 score is the harmonic mean of precision and recall, providing a single metric for correctness and completeness. This is particularly relevant to Cantus Database users, as maximizing precision ensures all results are accurate with no false positives, while high recall guarantees that no relevant items are omitted from the search. The final metric used is the average across the 45 queries.

4 Results

We evaluated the performance of three LLMs (GPT-o1, Claude, and Grok) in generating accurate SQL queries for the Cantus Database, comparing two scenarios: with and without a schema's predefined options. Results are reported as counts out of 45 for correct values with and without ordering, along with precision, recall, and F1 scores. Findings are shown in *Table 1* and *Table 2*.

Metric	GPT	Claude	Grok
Unordered (count)	**23**	**23**	16
Ordered (count)	**16**	15	11
Precision	0.70	**0.75**	0.56
Recall	0.69	**0.74**	0.47
F1	0.68	**0.70**	0.48

Table 1: Results for the schema without options. The unordered and ordered metrics display counts out of 45.

Metric	GPT	Claude	Grok
Unordered (count)	**32**	28	23
Ordered (count)	**23**	16	10
Precision	**0.92**	0.85	0.69
Recall	**0.88**	0.78	0.60
F1	**0.88**	0.78	0.61

Table 2: Results for the schema with options. The unordered and ordered metrics display counts out of 45.

5 Discussion

5.1 Performance on Schema Without Options

For schemas without options (*Table 1*), GPT and Claude achieve the highest unordered scores, each with 23 correct queries out of 45, while Grok trails with 16. For ordered results, GPT leads with 16 correct queries, followed by Claude with 15, and Grok with 11.

Precision, recall, and F1 scores provide a better understanding of the models' performance. While GPT and Claude generate the same number of correct unordered queries, Claude outperforms GPT in overall metrics, achieving a precision of 0.75, a recall of 0.74, and an F1 score of 0.70. This suggests that Claude's incorrect queries are generally closer to being correct. GPT follows closely with a precision of 0.70, a recall of 0.69, and an F1 score of 0.68. Grok lags behind both, with a precision of 0.56, a recall of 0.47, and an F1 score of 0.48.

5.2 Performance on Schema with Options

All models demonstrate improved performance when schemas include options (*Table 2*). GPT leads with the highest unordered count of 32 and ordered count of 23, followed by Claude with unordered and ordered counts of 28 and 16, respectively. Grok shows the weakest performance, with unordered and ordered counts of 23 and 10, respectively.

All models also show improved precision, recall, and F1 scores when provided with schema options. GPT achieved the most significant gains, leading with scores of 0.92 for precision and 0.88 for both recall and F1. Claude also improved, though less markedly, with scores of 0.85 for precision and 0.78 for recall and F1. Grok, while showing some progress, remained the weakest performer, with precision at 0.69, recall at 0.60, and F1 at 0.61.

5.3 Implications and Insights

The results indicate that GPT achieves the best outcomes when schema options are available, demonstrating its ability to leverage additional information to enhance performance. In contrast, Claude delivers stronger results without schema options, suggesting strength in handling more ambiguous contexts. Lastly, Grok consistently lags behind under both conditions.

5.4 Limitations

One limitation of this research is that the CantusDB user interface limits the kinds of queries that can be generated. Addressing this presents an interesting opportunity for future research. However, it would require more effort, as the SQL queries would need to be written manually instead of generated automatically by the website.

Another limitation is the reliance on metrics that evaluate query result matching rather than SQL correctness, potentially overlooking syntax validity, query complexity, and edge cases. Adding contextual relevance metrics could better assess how well generated queries reflect the intent of the natural language input, considering ambiguity and phrasing.

The methodology could also be improved with greater automation. Manual updates to data files and results are prone to errors and labor-intensive. Automating this process with scripts would minimize errors and enable future iterations to handle larger datasets efficiently.

Lastly, extraneous information adds noise and hinders performance, while missing details force models to make assumptions, leading to incorrect queries. Providing specific values in the prompt ensures the generated SQL queries align with the database. Future research is needed to refine the schema structure to optimize performance and better reflect user-based natural language queries.

6 Conclusion

This research confirmed the hypothesis that large language models can generate SQL queries for the specialized Cantus Database without fine-tuning. GPT achieved the best performance with schema options, correctly generating 32 out of 45 queries (ignoring order) and reaching an F1 score of 0.88. In the absence of options, Claude outperformed the other models, generating 23 correct queries and achieving an F1 score of 0.70. Grok consistently performed the weakest in both scenarios. The study underscores that incorporating select schema values with natural language queries significantly enhances performance across all models, demonstrating that modern large language models can address domain-specific Text-to-SQL tasks through prompt engineering alone. This efficient approach provides a viable alternative to fine-tuning, and future work could explore broader query types beyond the current limitations of Cantus Database.

7 Contributions

The project was a collaborative effort, with Zhanna leading the project setup, literature review, and JSON design. Lucas and Charles focused on schema generation and middleware, with Lucas also assisting with the Cantus Database setup and data extraction pipeline. Data generation was divided, with Zhanna handling GPT, Charles working with Grok, and Lucas managing Claude. Evaluation metrics were discussed by all, with Charles leading the analysis. Zhanna set up the LaTeX document and started the report, with all members contributing to every section. Regular meetings ensured clear tasks, progress, and revisions, highlighting strong teamwork.

References

Anthropic. 2023. Claude. Accessed: December 2024.

Cantus Team. 2024. Cantus: A database for latin ecclesiastical chant - inventories of chant sources. Accessed 2024.

Django Software Foundation. 2024. Django web framework. Version 4.2.16.

Docker, Inc. 2024. Docker. Version 27.x.

ECMA International. 2024. Javascript object notation (json). Data storage format.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-SQL empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.

OpenAI. 2024. Chatgpt-o1. `https://chat.openai.com/`. Accessed: December 2024.

PostgreSQL Global Development Group. 2024. Postgresql database system. Version 14.14.

Python Software Foundation. 2024. Python programming language. Version ^3.9.

N. Rajkumar, R. Li, and D. Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*. [Online]. Available: `https://doi.org/10.48550/arXiv.2204.00498`.

Richard Roberson, Gowtham Kaki, and Ashutosh Trivedi. 2024. Analyzing the effectiveness of large language models on text-to-sql synthesis. *arXiv preprint arXiv:2401.12379*.

xAI. 2024. Grok-2 beta release. Accessed: December 2024.