Understanding the Effects of Noise in Text-to-SQL: An Examination of the BIRD-Bench Benchmark

Niklas Wretblad[1,*] **Fredrik Gordh Riseby**[1,*] **Rahul Biswas**[2]
Amin Ahmadi[2] **Oskar Holmström**[1]
[1]Linköping University, [2]Silo AI
niklas.wretblad@liu.se

Abstract

Text-to-SQL, which involves translating natural language into Structured Query Language (SQL), is crucial for enabling broad access to structured databases without expert knowledge. However, designing models for such tasks is challenging due to numerous factors, including the presence of 'noise,' such as ambiguous questions and syntactical errors. This study provides an in-depth analysis of the distribution and types of noise in the widely used BIRD-Bench benchmark and the impact of noise on models. While BIRD-Bench was created to model dirty and noisy database values, it was not created to contain noise and errors in the questions and gold SQL queries. We found that noise in questions and gold queries are prevalent in the dataset, with varying amounts across domains, and with an uneven distribution between noise types. The presence of incorrect gold SQL queries, which then generate incorrect gold answers, has a significant impact on the benchmark's reliability. Surprisingly, when evaluating models on corrected SQL queries, zero-shot baselines surpassed the performance of state-of-the-art prompting methods. We conclude that informative noise labels and reliable benchmarks are crucial to developing new Text-to-SQL methods that can handle varying types of noise. All datasets, annotations, and code are available at this URL.

1 Introduction

Text-to-SQL with large language models facilitates broader access to structured databases without requiring expert knowledge. To develop such models, high-quality open datasets and benchmarks are essential resources, and over the years, several benchmarks and datasets have been created. Early benchmarks, such as WikiSQL (Zhong et al., 2017), modeled simple scenarios, often with single-table queries, and following datasets attempts to closer approximate real-world scenarios: complex queries with join-statements over several tables (Yu et al., 2018), unseen domain-specific datasets (Gan et al., 2021b; Lee et al., 2021), and noisy questions (Gan et al., 2021a). BIRD-Bench, a recent and challenging benchmark, aims to further close the gap between Text-to-SQL research and real-world applications by for example containing large and dirty database values and requiring external knowledge (Li et al., 2023).

Question

- What is the average loan amount by male borrowers?

Incorrect Gold Query

```
SELECT AVG(T3.amount) FROM client AS T1
INNER JOIN account AS T2 ON T1.district_id = T2.district_id
INNER JOIN loan AS T3 ON T2.account_id = T3.account_id
WHERE T1.gender = 'M'
```

Corrected Query

```
SELECT AVG(T1.amount) FROM loan AS T1
INNER JOIN account AS T2 ON T1.account_id = T2.account_id
INNER JOIN disp AS T3 ON T2.account_id = T3.account_id
INNER JOIN client AS T4 ON T3.client_id = T4.client_id
WHERE T4.gender = 'M'
```

Figure 1: Example of an incorrect SQL query that generates the wrong gold reference answer for the given question. The JOIN operation incorrectly matches clients and accounts by district_id. Due to the possibility of multiple clients and accounts in the same district, accounts are incorrectly associated with the wrong users.

While BIRD-Bench does not explicitly introduce noise to the questions in the data, it could be that it is added inadvertently due to human error during dataset creation. For the same reason, noise is an essential aspect of real-world use cases, as human inputs often are ambiguous and contain syntactical errors. However, for the benchmark to be a helpful tool for judging model properties, such as noise handling, the data must be valid and inform us in what areas a model can be improved.

This paper continues the tradition of examining the suitability and limitations of open datasets and benchmarks. We specifically focus on how noise is represented in questions and queries in BIRD-

[*]Equal Contribution

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 356–369

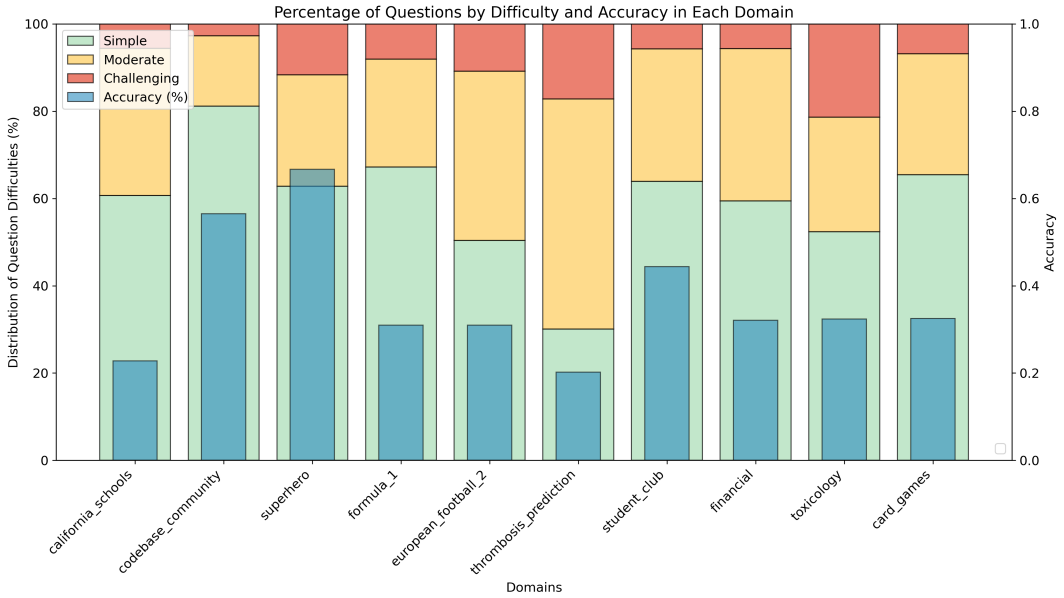

Figure 2: Distribution of question difficulties and execution accuracy of the DIN-SQL model on the different domains of the BIRD-Bench development set.

Bench. We perform a qualitative analysis of what types of noise exist in the data and the noise distribution in specific domains. We then study the effects of noise on different models and prompting techniques, using both strong baselines and state-of-the-art methods.

We find that noise in questions and gold SQL queries is prevalent, that noise is unevenly distributed across domains, and that categories of noise types are represented unequally in the data. Errors in gold SQL queries are also common and decrease the reliability of BIRD-Bench. When evaluating models on a dataset with corrected gold queries, the performance gap between zero-shot baselines and state-of-the-art prompting techniques is closed, questioning how we should interpret model performance on BIRD-Bench.

2 Related Work

Datasets WikiSQL is a large Text-to-SQL dataset containing only simple SELECT and WHERE operations without nested queries or JOIN operations (Zhong et al., 2017). SPIDER (Yu et al., 2018) was later developed to approximate real-life scenarios more closely, requiring models to construct complex queries and understand the database schema. While complexity is a critical aspect of real use cases, variations of SPIDER have been created to contain noisy questions (Gan et al., 2021a) and domain-specific questions (Gan et al., 2021b). BIRD-Bench was created to close the gap between academic research and real-world applications by introducing large and dirty database values, questions requiring external knowledge and optimizing SQL execution efficiency (Li et al., 2023).

Text-to-SQL Methods The notable gap in accuracy between automated systems (65.45%) and human experts (92.96%)[1], highlights the need for ongoing developments in Text-to-SQL models.

Different approaches have been taken to create models capable of Text-to-SQL generation. A more traditional approach is to finetune LLMs on Text-to-SQL examples. While these models offer promising results, there is a performance gap to instruction-tuned LLMs, in particular GPT-4, that is adapted to the Text-to-SQL task through prompt engineering (Li et al., 2023). Prompts are often chained, where each prompt is applied to the task sub-problems, such as schema linking, decomposition of queries, and refinement of model generations (Pourreza and Rafiei, 2023a; Wang et al., 2023).

Noise in Datasets The contemporaneous works of Wang et al. (2023) and Sun et al. (2024) shows that ambiguous questions and incorrect SQL queries exist in BIRD-Bench. However, unlike our work, they do not study how noise varies across domains or how the identified noise and errors affect

[1]BIRD-Bench benchmark as of 2024-02-16 (https://bird-bench.github.io)

Statistic	Financial	California Schools	Superhero	Toxicology	Thrombosis Prediction
Question & SQL query pairs with noise	52/106 (49%)	9/20 (45%)	3/20 (15%)	7/20 (35%)	8/20 (40%)
Noisy questions	44/106 (41.5%)	5/20 (25%)	2/20 (10%)	6/20 (30%)	3/20 (15%)
Erroneous gold queries	22/106 (20.7%)	8/20 (40%)	1/20 (5%)	2/20 (10%)	6/20 (30%)

Table 1: Statistics of the total amount of pairs of questions and SQL queries that contain errors and the amount of errors for questions and gold SQL queries separately across five domains.

Noise Type	Financial	California Schools	Superhero	Toxicology	Thrombosis Prediction
Spelling/Syntactical Errors	23	2	1	4	2
Vague/Ambiguous Questions	17	1	1	1	1
Incorrect SQL query	22	8	1	2	6
Synonyms	2	0	0	0	0
String Capitalization	7	0	0	0	0
Question does not map to DB	1	4	1	0	0
Total number of errors	72	15	4	7	9

Table 2: Distribution of different types of noise encountered in the domains.

model performance. Pourreza and Rafiei (2023b) perform a more fine-grained analysis of incorrect SQL queries but also mention categories of noise that we cover in our work (e.g., natural language question does not match database schema). In contrast to their work, we perform a more fine-grained analysis of noise in the natural language questions, for example the effects of syntactical errors, synonyms, and ambiguous questions.

Katsogiannis-Meimarakis and Koutrika (2023) points out that database schemas often misalign with data entities, which may cause lexical or syntactic ambiguities affecting Text-to-SQL models.

3 Method

3.1 Data

The BIRD-Bench dataset (Li et al., 2023) is studied in this paper as it is a recent and widely used dataset that is the most similar to real world scenarios among current benchmarks. BIRD contains 12,751 samples across many domains. Because of the time-consuming human annotation performed in this work, the main focus of the analysis is on the financial domain[2], which includes queries related to banking operations.

The development set of the financial domain contains 106 question and SQL query pairs, which represent approximately 7.5% of the data points in the development set, and are structured around eight distinct tables presented in full in Appendix A.1. Each question is annotated with a difficulty level (simple, moderate, and challenging). The specific distribution is found in Figure 2.

We selected four additional domains to validate our noise analysis of the financial domain and performed the same analysis on 20 randomly sampled questions from each domain. The domain selection was based on question difficulties and model accuracy of DIN-SQL[3], as presented in Figure 2. We selected *California Schools* with low accuracy and simple questions, *Superhero* with high accuracy and simple questions, *Toxicology* with similar accuracy to the financial domain but more complex questions, and *Thrombosis Prediction* with low accuracy and moderately difficult questions.

3.2 Annotation of Noise

All questions and SQL queries in the selected domains were annotated to determine whether they contained errors. The annotations were performed independently by two authors of this paper, fluent in English and experts in SQL. In the first phase, annotators independently identified questions and SQL queries with errors. The Cohen's Kappa coefficient was 0.73, demonstrating a substantial level of agreement between annotators. The annotators then independently named the types of errors. In the second phase, the annotators resolved disagreements by observing the other annotator's reasoning and the remaining disagreements were

[2]This was also motivated by the fact this paper was a collaborative endeavor with the Swedish bank SEB.

[3]Results of DIN-SQL across domains were provided by the creators of DIN-SQL.

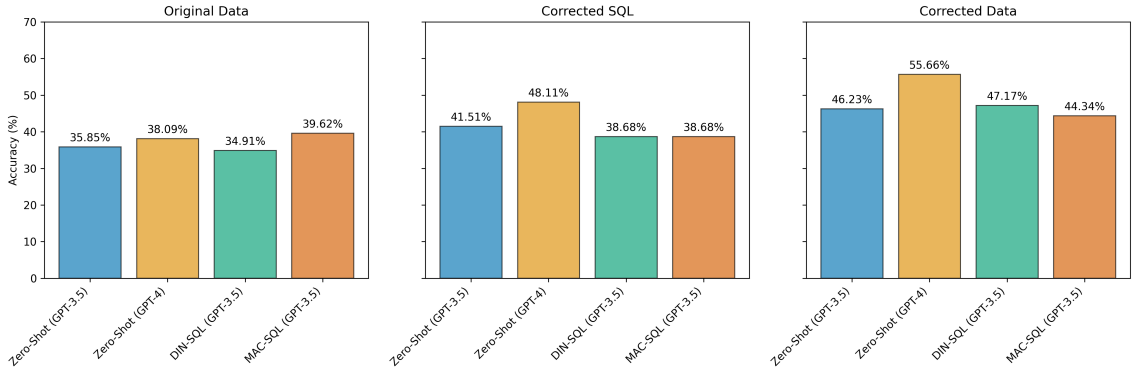

Figure 3: Accuracy of various models on Bird-Bench's financial domain. Models are evaluated on the original data (left), corrected SQL queries (middle), and corrected SQL queries and corrected noisy questions.

resolved through discussion. The identified errors were grouped based on similarity and named after the errors' common properties, as shown in Table 2. The annotations were then used to generate two distinct datasets: one where SQL was corrected, and one where both SQL queries and noisy questions were corrected.

3.3 Models and Prompt Techniques

Two models, GPT-3.5 and GPT-4, were used with three different prompting methods: zero-shot prompting as a baseline and the more advanced DIN-SQL (Pourreza and Rafiei, 2023a) and MAC-SQL (Wang et al., 2023). We used GPT-3.5 and GPT-4 for zero-shot prompting, but for the advanced prompting techniques, we only used GPT-3.5 since chaining prompts with GPT-4 was beyond the resources for this project. We chose the models and prompting methods because they were the highest-performing publicly available models on BIRD-Bench at the time of writing.

Information about the database schema is crucial to generating correct queries for BIRD-Bench questions. DIN-SQL and MAC-SQL has a predefined format for adding the database schema. For the zero-shot model, we provide the database schema in-context in the form of SQL table creation statements, as this has been shown to improve accuracy compared to other formats (Nan et al., 2023). The prompt template for the zero-shot model is found in Appendix A.2. The code base is published after the anonymity period.

4 Qualitative Analysis of Noise

Even though BIRD-Bench was not intentionally created to contain noise in questions and SQL queries, our analysis reveals that noise exists in all studied domains to different extents. The financial domain exhibits the highest levels of noise at 49% closely followed by the *California Schools* domain at 45%, as shown in Table 1. In contrast, the *Superhero* domain demonstrated the lowest noise levels, with only 15% of data points containing errors. As presented in Section 3.1 and Figure 2, the *Superhero* domain had the highest accuracy while having a similar distribution of question difficulties. This could indicate that model accuracy across tasks correlates with noise, which implies that noise in questions and SQL queries need to be carefully considered during dataset design.

The categories and absolute frequency of noise per dataset are presented in Table 2, and both examples and descriptions of the noise types are presented in Appendix A.3. Our analysis shows that spelling/syntactic errors and incorrect SQL queries were most prevalent in the financial domain. The presence of noise in questions is not necessarily undesirable, as it more closely mimics real-life scenarios. However, noise distribution across the categories is unequal. While this could approximate a real-world distribution, it might unfairly bias the benchmark towards models better at handling syntactical errors. Given the uneven distribution of errors and the lack of noise labels, the benchmark does not inform which noise types are challenging for current models and in which areas they should improve.

A more severe issue is that all domains contained incorrect SQL queries, which are used for generating gold reference answers. An example of an erroneous SQL query is shown in Figure 1. These types of errors question the reliability of the benchmark to accurately determine model performance, which is explored in the next section.

Error Category	Total	DIN-SQL (3.5)	Zero-shot (3.5)	Zero-shot (4)	MAC-SQL (3.5)
Spelling/Syntactical Errors	23	2	6	4	6
Vague/Ambiguous Questions	17	1	2	3	4
Incorrect SQL	22	0	2	2	4
Synonyms	2	0	0	0	0
String Capitalization	7	2	1	1	0
Question does not map to DB	1	0	0	0	0

Table 3: Model performance on the financial domain for various error categories and overall correct predictions on non-erroneous questions.

5 Impact of Noise on Model Performance

We apply models to the original dataset, a dataset where SQL has been corrected, and a dataset where both SQL queries and noisy questions have been corrected. Figure 3 presents the results of a single evaluation for all models on all datasets.

MAC-SQL slightly outperforms DIN-SQL and the zero-shot baselines on the original dataset, where noise exists in both questions and queries. However, correcting SQL queries decreases MAC-SQL's performance, tying it with DIN-SQL as the poorest performers. Surprisingly, even the zero-shot GPT-3.5 baseline outperforms the more advanced DIN-SQL and MAC-SQL. The dataset with corrected SQL queries could also be considered optimal since gold labels are correct and noise in questions is represented. Given the drastic re-ranking of models, it is relevant to question if BIRD-Bench is a reliable assessor of models and a useful tool to assist researchers in developing new methods for Text-to-SQL.

When evaluating models on the dataset with both questions and SQL queries corrected, the accuracy of all models increases significantly. While zero-shot GPT-4 performs the best, the remaining models perform similarly with DIN-SQL slightly ahead. Compared to the ideal scenario where only SQL queries are corrected, the presence of noise noticeably impacts all models' accuracy. However, models are not equally affected by noise as some models have a more pronounced increase in accuracy. Table 3 presents each model's performance for the error categories. MAC-SQL outperforms the other models slightly on errors related to Spelling and Syntactical Errors, Ambiguous Questions, and Incorrect SQL. The main difference between MAC-SQL and the other methods is an extensive filtering process of tables and columns and the increase of relevant information in the context could make the model more robust to noise. However, such a hypothesis must be confirmed or rejected by studying what the model has seen during the generation phase, which we leave to future studies.

6 Conclusions and Future Work

This paper analyzed the quality and distribution of noise in the BIRD-Bench benchmark for Text-to-SQL. We show that noise in both questions and SQL queries are prevalent, and noise is unevenly distributed across noise types and domains. Errors in gold SQL queries were common, decreasing the reliability of BIRD-Bench. Surprisingly, when evaluating models on corrected gold queries, zero-shot baselines surpassed more advanced prompting techniques. These findings highlight the necessity for developing benchmarks that can guide researchers in designing models that are more resistant to noise. Therefore, a significant improvement would be to label noise types across the dataset. In future work, we plan to study how large language models can be applied to noise classification, a new task that could also be critical in systems where Text-to-SQL is employed.

Overall, this study provides a deeper understanding of how noise is expressed in Text-to-SQL tasks and how noise and models interact, pinpointing areas for improvement in the BIRD-Bench dataset.

Limitations

While our study provides valuable insights regarding the influence of dataset noise in Text-to-SQL translation tasks, it has several limitations. As the analysis was performed mainly on the BIRD-Bench dataset's financial domain, our findings' generalizability may be limited. We only examined a small subset of other domains to validate our findings, which may represent only some of the noise distribution across domains.

Additionally, annotators may have introduced subjective bias during noise annotation, even

though we attempt to minimize this by having two independent annotators. Further, our decision to categorize noise into six specific classes might have oversimplified the complexity and diversity of noise types in these benchmarks.

Our choice of models and prompting techniques could also be a potential limitation. We only employed two models, GPT-3.5 and GPT-4, and three different prompting methods. Evaluating a more comprehensive array of models and prompting techniques might have given a more comprehensive understanding of their performance under the influence of noise.

Lastly, the substantial effort required to correct SQL queries and noisy questions in the dataset may have introduced errors despite the review process. This might influence the model performances we report when evaluating models on the corrected datasets.

Acknowledgments

We extend our gratitude to Mohammadreza Pourreza for the results from the DIN-SQL model. We are also grateful to SEBx for their generous support and the provision of resources. Additionally, this research was partial funded by the National Graduate School of Computer Science in Sweden (CUGS).

References

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021a. Towards robustness of text-to-SQL models against synonym substitution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2505–2515, Online. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021b. Exploring underexplored limitations of cross-domain text-to-SQL generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8926–8931, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

George Katsogiannis-Meimarakis and Georgia Koutrika. 2023. A survey on deep learning approaches for text-to-sql. *The VLDB Journal*, 32(4):905–936.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic evaluation of text-to-SQL parsers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2261–2273, Online. Association for Computational Linguistics.

Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. In *Advances in Neural Information Processing Systems*. Spotlight Poster.

Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023. Enhancing text-to-SQL capabilities of large language models: A study on prompt design strategies. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14935–14956, Singapore. Association for Computational Linguistics.

Mohammadreza Pourreza and Davood Rafiei. 2023a. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. In *Advances in Neural Information Processing Systems 36*. Accepted for poster presentation, full citation details to be updated.

Mohammadreza Pourreza and Davood Rafiei. 2023b. Evaluating cross-domain text-to-SQL models and benchmarks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1601–1611, Singapore. Association for Computational Linguistics.

Ruoxi Sun, Sercan Ö. Arik, Alex Muzio, Lesly Miculicich, Satya Gundabathula, Pengcheng Yin, Hanjun Dai, Hootan Nakhost, Rajarishi Sinha, Zifeng Wang, and Tomas Pfister. 2024. Sql-palm: Improved large language model adaptation for text-to-sql (extended).

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Qian-Wen Zhang, Zhao Yan, and Zhoujun Li. 2023. Mac-sql: A multi-agent collaborative framework for text-to-sql.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *arXiv e-prints*, page arXiv:1709.00103.

A Appendix

A.1 Database Schema of the Financial Domain

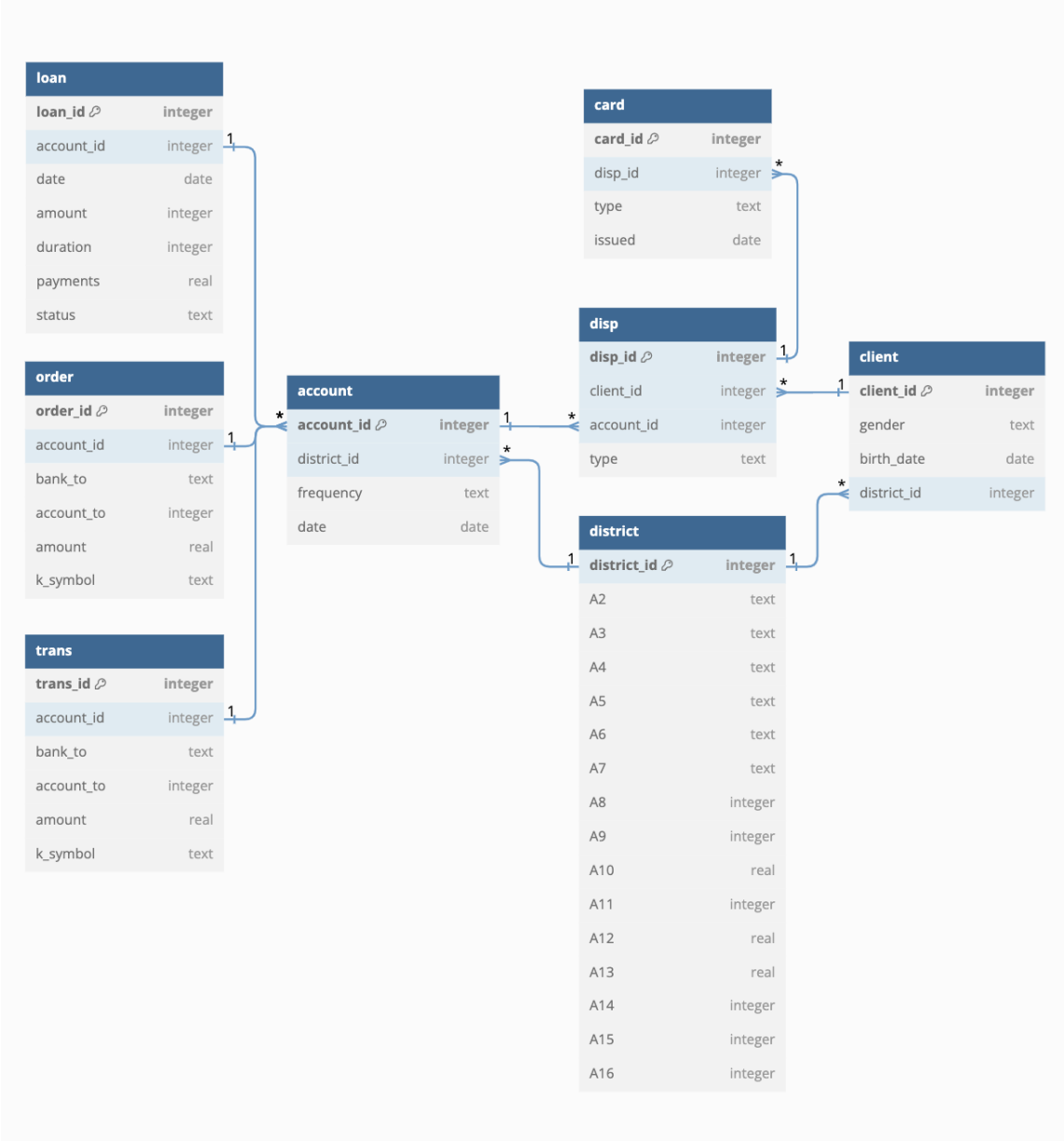

Figure 4: Database schema of the database in the financial domain of BIRD-Bench.

Figure 4 displays the database schema for the financial domain. This schema contains various tables, such as those for loans, transactions, accounts, cards and clients, all reflecting the financial orientation of the database. Descriptions of what information these tables contain are presented in Table 4. The database consists of 55 columns distributed across eight distinct tables. While the majority of the column names are intuitively understandable, some present interpretative challenges, as evident in the schema. An illustrative example is the district table, which incorporates 16 unique columns. This includes a column titled *district_ID* along with 15 other columns, ranging from *A2* to *A16*. The latter columns' names do not readily convey the nature of the data they hold, making them less intuitive to understand. In practice, a database schema will often be accompanied by a data dictionary or documentation that explains each table and column in detail. Such documentation would typically provide the context needed to fully understand the meaning of each element in the schema, the range of possible values for fields with unspecified types, and the business logic underlying the relationships. Without this additional documentation, fully

interpreting and effectively using the database can be challenging as illustrated by the column names in the districts table. The BIRD-Bench dataset includes a unique feature for each question termed *hint*. This feature is designed to offer insights or supplementary information corresponding to the specifics detailed in such database documentation. This feature is provided to all models described in 3.3 for each question during the experiments.

Table 4: Table descriptions of the tables in the database of the financial domain of BIRD-Bench.

Table Name	**Description**
loan	Contains details of loans.
order	Holds information about monetary orders.
trans	Represents financial transactions.
account	Contains account information.
disp	Links clients to accounts (dispositions).
card	Contains details about cards issued.
client	Holds client information.
district	Contains details about districts or regions.

Further, the lines in Figure 4 between the tables represent relationships, where the nature of the relationship is indicated by the shape of the tail end of the lines where they connect to each table. A one-to-many relationship is indicated by the line beginning with a single line and the one digit above it, and then ending in a crow's foot (three lines) at the opposite end. For example, an account can have multiple orders, transactions, dispositions, and loans associated with it, but each of those entities is only linked to one account. An account can have many loans, but one loan is exclusively only linked to one account, which makes sense. Further, clients and accounts are related through the disposition table in a many-to-many relationship. An account can have many different clients associated with it, for example, one client listed as the owner of the account and multiple other clients listed as users for the account. This could for example be practical for sharing an account in a family, where one parent could be the owner of the account and then multiple other family members listed as users. A single client can also be related to many different accounts in the other way around.

A.2 Prompt Templates

```
"""Database schema in the form of CREATE_TABLE statements:

{database_schema}

Using valid SQL, answer the following question based on the
tables provided above.

Hint helps you to write the correct sqlite SQL query.
Question: {question}
Hint: {evidence}
DO NOT return anything else except the SQL query."""
```

Listing 1: Zero-Shot Prompting Template.

The prompt template underlying the zero-shot models described in Section 3.3 can be found in Listing 1. The prompt integrates a given question, the associated database schema, an instruction directing the LLM to generate valid SQL, and a hint provided by the BIRD-Bench dataset. The hint is designed to offer

insights or supplementary information needed in order to accurately interpret the database schema and to correctly convert the question into a SQL query. Note that the other models implemented in this research is also provided with this feature.

A.3 Examples of Errors and Corrections

This section provides examples of erroneous data points and their corrections from the different error categories found in Table 1.

Example 1: Spelling/Syntactical Error

In Figure 5, an example question with a syntactical error is provided, representing the question with ID 125 from the financial domain in the BIRD-Bench development set. The grammatical structure of the question complicates the interpretation of its meaning for a human reader and makes it difficult to understand which information it is asking for. Therefore, there is a chance that an LLM might also misinterpret the question. A corrected version of the question can be seen in the figure.

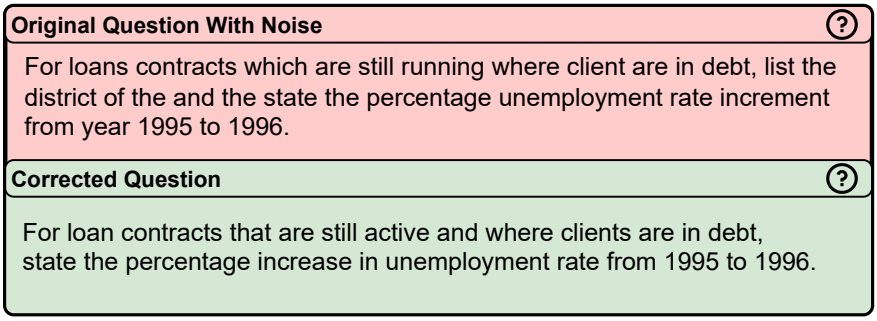

Figure 5: Question with ID 125 from the development set of BIRD-Bench which contains syntactical errors and a corrected version of the question.

Example 2: Ambiguous/Vague Question

Figure 6 displays the data point with ID 159 from the financial domain of the development set of BIRD-Bench. It contains an error which were grouped into the ambiguous/vague question category. The challenge lies in the natural language question's ambiguity, specifically in the phrase *"List all the withdrawals..."* This ambiguity revolves around determining which columns to return when executing the SQL query.

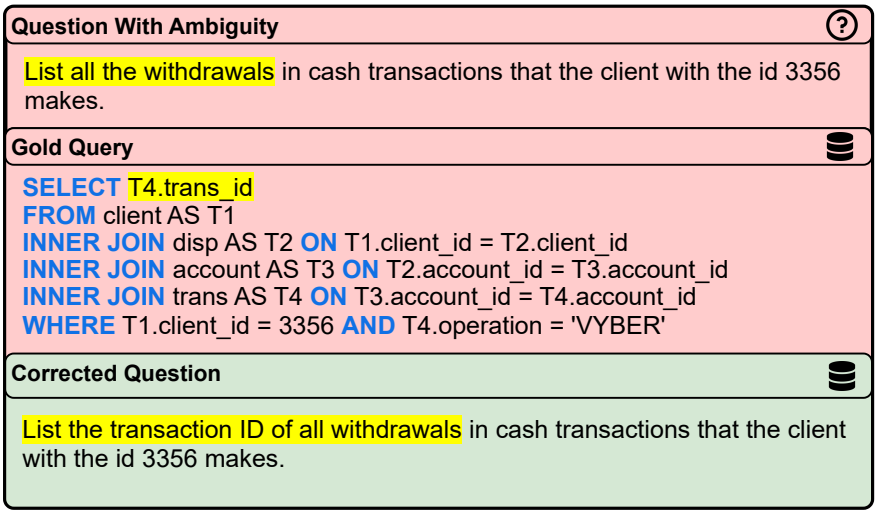

Figure 6: Question, gold SQL query and a corrected version of the question corresponding to the data point with ID 159 from the development set of BIRD-Bench, showcasing an error in the ambiguous/vague category.

Example 3: Incorrect Gold SQL

Figure 7 showcases an incorrect golden SQL query found in the data point with ID 132 of the financial domain of the development set of BIRD-Bench. The JOIN operation incorrectly matches clients and accounts by district_id. Due to the possibility of multiple clients and accounts in the same district, accounts are incorrectly associated with the wrong users.

Question

- What is the average loan amount by male borrowers?

Incorrect Gold Query

```
SELECT AVG(T3.amount) FROM client AS T1
INNER JOIN account AS T2 ON T1.district_id = T2.district_id
INNER JOIN loan AS T3 ON T2.account_id = T3.account_id
WHERE T1.gender = 'M'
```

Corrected Query

```
SELECT AVG(T1.amount) FROM loan AS T1
INNER JOIN account AS T2 ON T1.account_id = T2.account_id
INNER JOIN disp AS T3 ON T2.account_id = T3.account_id
INNER JOIN client AS T4 ON T3.client_id = T4.client_id
WHERE T4.gender = 'M'
```

Figure 7: Example of an incorrect SQL query that generates the wrong gold reference answer for the given question. The JOIN operation incorrectly matches clients and accounts by district_id. Due to the possibility of multiple clients and accounts in the same district, accounts are incorrectly associated with the wrong users.

Example 4: Synonyms

Figure 8 demonstrates how specific wordings can complicate interpretation for an LLM. The term *sum*, being both a SQL keyword and a descriptor, led to the LLM's literal interpretation and the incorrect summation of a transaction. The actual intent was to inquire about the transaction's balance or amount. A rephrased question resulted in the LLM generating the correct SQL query, fetching the intended information, as seen in the figure.

Original Question With SQL Keyword/Synonym

What is the sum that client number 4's account has following transaction 851? Who owns this account, a man or a woman?

Predicted Query (Incorrect)

```
SELECT
  SUM(trans.amount)
  client.gender
FROM trans
JOIN account ON trans.account_id = account.account_id
JOIN disp ON account.account_id = disp.account_id
JOIN client ON disp.client_id = client.client_id
WHERE trans.trans_id = 851 AND client.client_id = 4;
```

Corrected Question

What is the balance of client number 4's account following transaction 851? Who owns this account, a man or a woman?

Predicted Query (Correct)

```
SELECT
  trans.balance
  client.gender
FROM
  trans
  JOIN disp ON trans.account_id = disp.account_id
  JOIN client ON disp.client_id = client.client_id
WHERE trans.trans_id = 851 AND client.client_id = 4;
```

Figure 8: Question from data point with ID 177 from the development set of BIRD-Bench containing a difficult synonym, a corrected version of the question with the synonym replaced and corresponding predicted SQL queries by the DIN-SQL (GPT-3.5) model described in Section 3.3. Showcases the difficulty of synonyms on model predictions.

Example 5: String Capitalization

As a consequence of SQL being a case-sensitive language when comparing string values in a query, the way a question is formulated regarding the use of uppercase or lowercase letters when asking for a specific value affects the result. This is because the LLM will most likely use the specific entry as given when generating the query, unless it has knowledge of the case used for different entries in the database. Therefore, in Figure 9, an example is provided where the terms "East" and "North" are mentioned with initial capital letters, as is commonly the case. However, the entries for these column values are in lowercase in the database, which means the question needs to account for this for the LLM to be able to generate a correct query. The corrected question and the SQL query generated from it can also be seen in Figure 9.

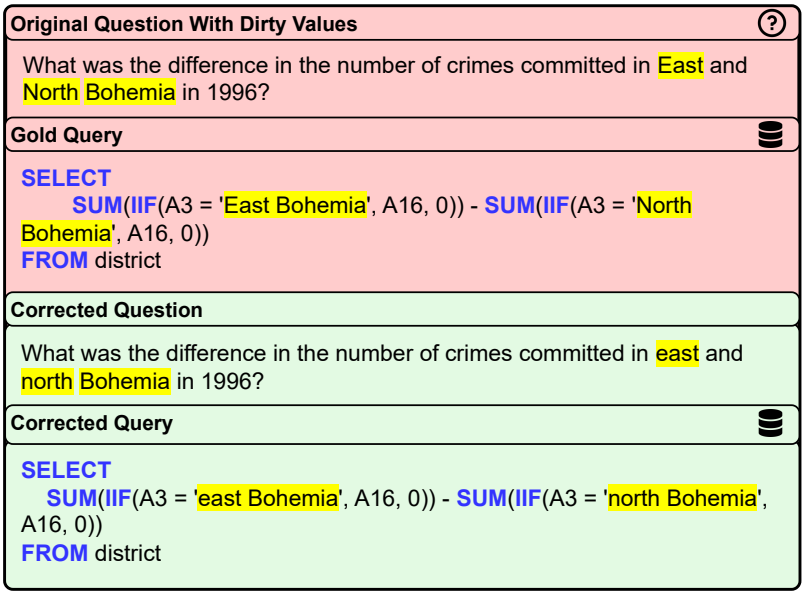

Figure 9: Example Ambiguous.

Example 6: Database Schema Non-Alignment

Incorrect Question	Description
What is the disposition ID of the client who made $5100 USD transaction on 1998/9/2?	The question asks for a single disposition ID, which does not reflect that there is a one-to-many relation between client and disposition, and most likely it won't be possible to return a single ID.
List out the account numbers of clients who are youngest and have highest average salary?	There is no information about salaries of specific clients in the database.

Table 5: Examples of questions that does not map to the database schema and accompanying descriptions of why they do not.