Data Ambiguity Strikes Back: How Documentation Improves GPT's Text-to-SQL

Zezhou Huang
Columbia University
zh2408@columbia.edu

Pavan Kalyan Damalapati
Columbia University
pd2720@columbia.edu

Eugene Wu
DSI, Columbia University
ewu@cs.columbia.edu

Abstract

Text-to-SQL allows experts to use databases without in-depth knowledge of them. However, real-world tasks have both query and data ambiguities. Most works on Text-to-SQL focused on query ambiguities and designed chat interfaces for experts to provide clarifications. In contrast, the data management community has long studied data ambiguities, but mainly addresses error detection and correction, rather than documenting them for disambiguation in data tasks. This work delves into these data ambiguities in real-world datasets. We have identified prevalent data ambiguities of value consistency, data coverage, and data granularity that affect tasks. We examine how documentation, originally made to help humans to disambiguate data, can help GPT-4 with Text-to-SQL tasks. By offering documentation on these, we found GPT-4's performance improved by 28.9%.

1 Introduction

Text-to-SQL is widely used as it allows domain experts who aren't familiar with database structures or SQL to access data. Although specialized models have been developed and show promising results [36, 32], recent studies have found that, by increasing the size of both the model and training data, general-purpose Large Language Models (LLMs) like GPT-4 with around 1.7T parameters can achieve state-of-the-art performance [14, 19, 9] in Text-to-SQL tasks using the Spider benchmarks [35].

Unlike the Spider dataset, which is characterized by its well-structured schema and clean data, real-world Text-to-SQL tasks often present challenges due to ambiguities both from query and data:

- **Query Ambiguity**: The queries provided by domain experts can be interpreted in multiple ways with respect to the data. Common query ambiguities include query term holding multiple meanings [33, 36, 32, 36, 41], or the output schema being under-specified [9, 38].
- **Data Ambiguity**: The real-world concepts encapsulated within the data can be interpreted differently. Data ambiguity is a fundamental aspect of data, independent of the queries or the Text-to-SQL tasks at hand, and has been studied by the data management community for decades. This paper follows the scope of data ambiguities established by previous works [2, 13], which includes value consistency [22, 28] (e.g., do the values follow consistent formats?), data coverage [30, 25] (e.g., which subset of data this table covers?), data granularity [29, 7] (e.g., does each row records one event or an aggregation?), to more domain-specific column understanding [10, 18].

These types of ambiguities present a major challenge in any data task, including Text-to-SQL, because the LLM needs to contend with both the translation work *and* correctly interpreting the query and

Table Representation Learning Workshop at NeurIPS 2023.

data semantics. Unfortunately, data ambiguity is relatively unexplored, particularly in the domain of Text-to-SQL. Prior works study the model sensitivity to query ambiguity by artificially introducing ambiguous terms into queries [33, 38], and suggest solutions like consulting domain experts via chat interfaces [36, 41]. However, these approaches assume clean databases, which is often not true. Additionally, the user submitting the natural language query is frequently not the data provider and may not understand the subtle assumptions and semantics of the dataset. In contrast, the data management community has long studied data quality issues, but mainly in terms of detecting and fixing data errors [6, 23, 27] rather than documenting the dataset in a way that can disambiguate its application to different data tasks. We believe that LLMs, such as GPT-4 with strong general knowledge [24], offer an opportunity for data providers to document their datasets in natural language. While there have been some proposals to better document datasets [31, 4, 16], these standards are designed for human understanding and haven't seen wide adoption for general datasets. Whether LLMs can take advantage of documentation to improve data tasks remains an open question. There are other works that address ambiguity for Text to SQL [3, 11] by, e.g., letting the model search through perturbed SQLs to detect structural errors like wrong selection orders or missing joins. The paper concentrates on how documentation can resolve ambiguities inherent to the data itself.

In this paper, we study how combinations of data and query disambiguation work in isolation and together to improve Text-to-SQL tasks. We simulate a scenario where a data provider documents their data offline, and a user uses natural language to disambiguate their text input online. To delve into ambiguities in real-world datasets, we use KaggleDBQA, a Text-to-SQL benchmark collected from 8 real-world Kaggle databases with 18 tables. This benchmark had annotators draft 272 natural language queries, and SQL experts provided one SQL answer per query. KaggleDBQA provides basic data documentation to describe obscure column names. However, data ambiguity in KaggleDBQA goes beyond obscure column names and encompasses common data ambiguity issues, thus making it an intriguing subject for study. We illustrate these ambiguities with an example.

Example 1. Consider the database from KaggleDBQA that records football matches and betting data:

```
betfront: year, datetime, country, competion, match, home opening,..., away closing
football_data: season, datetime, div, country, league,..., bwd, bwa
```

Given the natural language query *"Which year has the most matches?"*, there are both query and data ambiguities:

- **(Query) Term Ambiguity: [33, 36, 32, 36, 41]** football_data uses season to represent time and season could span two years. Which year is the query asking for? Is it the start year or the end year?
- **(Query) Output Schema [9, 38]**: What's the expected schema of the output? Is it solely the 'year', or should it also provide the count of matches as the evidence?
- **(Data) Value Consistency [22, 28]**: How are matches formatted? Do they consistently follow the "teamA vs. teamB" format? This will influence the method of selecting matches.
- **(Data) Data Converage [30, 25]**: Do both tables contain every match? Do they contain mutually exclusive subsets, or do they intersect? This brings up the potential need for union or deduplication operations.
- **(Data) Data Granularity [29, 7]**: Does each row in a table correspond to a unique match, or can there be repeated rows for the same match due to updates in statistics or data as time progresses? This determines if a simple COUNT(*) would suffice or if we need to account for duplications with COUNT(distinct match)

Note that the benchmark's provided SQL answer (`SELECT YEAR FROM betfront GROUP BY YEAR ORDER BY count(*) DESC LIMIT 1`) makes several assumptions to clarify the aforementioned issues: each row in betfront is a unique match, and betfront contains all matches without the need to use football_data.

---

2 Disambiguation Methods

To control data and query ambiguity, we introduce methods to disambiguate data and query.

2.1 Data Disambiguation

We emulate a situation where data providers offer offline documentation to disambiguate data. We first consider the documentation used by previous works to help LLM understand data:

- **Schema**: Previous Text-to-SQL [19, 9] and popular open-source projects like Langchain [5] and LlamaIndex [20] only provide the schema. This can be ineffective for noisy data.

Table 1: Types of documentations for Data Disambiguation, and Query Disambiguation Methods

Type	*Example*
Name Description	*"bwd means Bet&Win draw odds."*
Value Consistency	*"Matches are consistently denoted in the format of 'home team - away team', for example, 'Malta - Albania'. There are no outliers."*
Data Converage	*'football_data' covers all the matches only from 2009-2013."*
Data Granularity	*"Each row in 'betfront' reports for each unique match in each competition, the detailed time, location and betting records. It is not aggregated."*
Term Ambiguity	*"In which year did the most matches take place?"*
Output Schema	*"The output must only contain the year."*

- **Name Description**: Prior works including KaggleDBQA [39, 15, 17] provide documentation to describe the meanings of the obscure column names.

- **Sample**: Data samples [14, 26, 37, 12] are provided to help GPT interpret the data. By default, we provide each table's sample of the first 5 rows.

However, none of the above tackle the data ambiguities detailed in Example 1. In response, we provide documentation for three common data ambiguity issues for KaggleDBQA. We disambiguate and document data by exploring data and interpreting the provided SQL answers:

- **Value Consistency [22, 28]**: For each column, we document whether the data is represented consistently in some formats. We also specify any outlier formats that exist.

- **Data Coverage [30, 25]**: We document the coverage of each table, specifying whether it represents the entirety of real-world events or if it has been subsetted in certain ways (e.g., time/location).

- **Data Granularity [29, 7]**: For every table, we document whether the rows represent aggregated data by some group-by keys, or raw data entries for some real-world events.

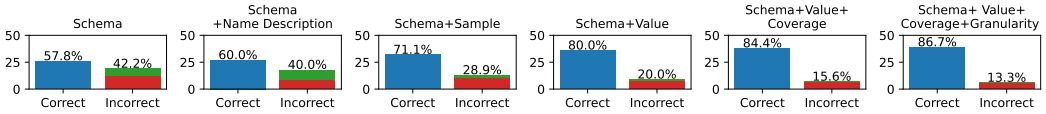
We provided the example documentation for Example 1 in Table 1.

Limitations: There are other common types of documentation we've not provided due to challenges in determining the ground truth. For example, half of the columns in "football_data" have $> 30\%$ missing values. It remains ambiguous whether these indicate non-applicability, data collection errors, or censoring [1]. KaggleDBQA doesn't address these missing values. In the absence of a reliable ground truth, we abstain from documenting them and leave them for future research.

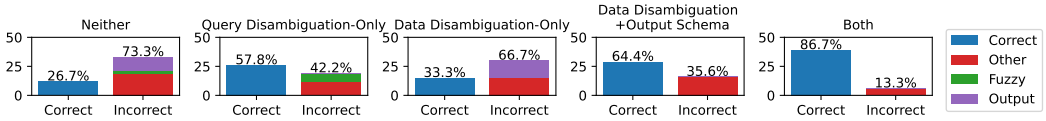
Levels of Documentation and Refinement: We vary documentation levels, beginning with the schema and then incrementally incorporating samples, Name descriptions, value consistency, data coverage, and data granularity. However, we observed that certain documentations are repetitive. Naively adding more adversely impacts GPT-4, because lengthy and irrelevant prompts hinder the LLM from focusing on the useful information [21, 8]. Therefore, we refine documentation in two ways: (1) Documentation describing each column (name description and value consistency) tends to be lengthy. We employ an agent approach [20, 5] first to let GPT-4 select up to 5 columns. Then, documentation is provided only for these. (2) Name description, sample, and value consistency have many overlaps, as they similarly help GPT understand the columns. We therefore only provide one.

2.2 Query Disambiguation

We disambiguate queries in two ways: (1) **Output Schema:** Almost all queries within KaggleDBQA have underspecified output schemas [9, 38]. We explicitly specify the output schema for all queries. (2) **Term Ambiguity:** Some queries also have ambiguous terms [33, 36, 32, 36, 41]. For instance, some queries ask about the "most dangerous places" without explaining what "dangerous" means. Other queries ask for crimes in "Manchester", which could refer to Greater Manchester or the city of Manchester. We carefully review each query and refine these terms based on the provided answers.



(a) Accuracy when the queries have been disambiguated, but the levels of documentation vary.



(b) Accuracy when the queris and data are disambiguated in isolation or together.

Figure 1: GPT-4 Error analysis. Blue bars are for correct, while others are for distinct types of errors.

3 Experiments

Data and Model: Due to the manual nature of disambiguation, we evaluate 2 (out of 8) KaggleDBQA databases (Soccer and Crime) with 45 queries. We use GPT-4 model with 8K context size. We employ the standard chain-of-thought to enhance GPT's performance and interpretability [34].

Evaluation Setting: In line with previous studies [17, 35], we assess SQL queries using exact match accuracy. It's possible for GPT-4 to produce semantically the same but syntactically different queries; we manually evaluate the queries to ensure that this doesn't occur. Despite our efforts to resolve ambiguities using the provided SQL answers (Section 2), we observed that 22.2% of them have errors: (1) *13.3% have inconsistencies*: We note variations in the interpretation of terms. E.g., for queries asking "area", some answers use the "lsoa" column (Lower Layer Super Output Areas), while others opt for the "location" (street-level). For consistency, we fix SQL answers to be consistent with the "lsoa" interpretation. (2) *8.9% have syntax errors*: We detect errors of missing 'Distinct' in count and improper null checks (= "" instead of "is Null"). We fix these syntax errors in SQL answers.

Error Analyses: We highlight two common mistakes GPT-4 made: (1) Adding extra columns to the output(`Output`) [9]. (2) Using exact string matches instead of fuzzy ones for selection (`Fuzzy`). If the only mistake GPT-4 makes falls into these two categories, we'll highlight them. For any other errors or if there are additional mistakes, we label them as `Other`.

3.1 Levels of Documentation for Data Disambiguation

We first disambiguate queries, and study how varying levels of documentation help Text-to-SQL.

Results: Figure 1a shows the results. (1) We find that GPT-4 achieves a high accuracy of 57.8% with only schema. Most errors arise due to the preference for exact string matching over fuzzy matching. (2) In contrast, only a 2.2% improvement is observed when `Name Description` is provided. We find that GPT-4 has the capability to infer full names from vague column names. KaggleDBQA documentation doesn't significantly aid GPT-4. (3) Giving samples helps GPT-4 avoid most `Fuzzy` errors. (4) By providing documentation on Value Consistency, GPT-4 can better avoid `Fuzzy` and apply correct predicates. E.g., with only samples, GPT-4 misinterprets "season" in $football\_data$ as only in the format of "Year1/Year2". By providing documentation specifying that "season" also contains a single year, GPT-4 fixes selection errors. (5) Data Coverage helps GPT-4 avoid mistakes in unioning and joining the "betfront" and "football_data" tables, as it understands a single table is sufficient for the query. (6) Data Granularity assists GPT-4 in applying predicates. E.g., one query asks for "street" crimes. GPT-4 previously misunderstood one row as one street crime. By specifying the row granularity as a crime on streets, roads or avenues, GPT-4 correctly refines the selection.

3.2 Compare Query vs Data Disambiguation

We assess the effects of query disambiguation (original, or with term and output schema disambiguated) versus data disambiguation (schema only, or schema with value, coverage, and granularity).

Results: Figure 1b shows the results. (1) When only the schema and the original query are provided, the accuracy is 26.7% and matches the KaggleDBQA results. (2) Query Disambiguation is pivotal: Replacing the original queries with disambiguated ones elevates the accuracy to 57.8%. (3) If we only disambiguate data but not query, accuracy is only 33.3%.While this might suggest that data

disambiguation by itself isn't as effective, we discover that 33.3% of the errors come from the output schema, which is easy to fix. (4) To verify this, we specify output schema (not term disambiguation) for queries, and the accuracy surges to 64.6%. This underlines the importance of data disambiguation. (5) Finally, disambiguating both yields the highest accuracy at 86.7%. 13.3% errors remain even with both documentation and query disambiguation. We investigate these and find that they are from domain-specific nuances. E.g., "home losing odds" corresponds to "away winning odds", but GPT-4 chooses "home winning odds". Addressing these needs domain-specific documentation [40].

4 Conclusion

This work studies ambiguities in real-world datasets and assesses how documentation aids GPT-4 in enhancing Text-to-SQL. Our findings reveal that data ambiguities are prevalent, and extend beyond obscure column names to issues like value consistency, data coverage, and granularity. By providing documentation on these issues, GPT-4's accuracy is improved by 28.9%. Looking forward, we intend to (1) investigate other data ambiguity issues, such as missing values, and (2) explore semi-automating the documentation process by leveraging GPT-4 to assist data providers.

Acknowledgements

This work was funded by the NSF under Grant Numbers 1845638, 2008295, 2106197, 2103794, 2312991, and was further supported by the Google PhD Fellowship, along with contributions from Amazon and Adobe. We are grateful to the Microsoft Gray Systems Lab for their assistance in providing us with access to compute resources.

References

[1] Alan C Acock. Working with missing values. *Journal of Marriage and family*, 67(4):1012–1028, 2005.

[2] Carlo Batini, Cinzia Cappiello, Chiara Francalanci, and Andrea Maurino. Methodologies for data quality assessment and improvement. *ACM computing surveys (CSUR)*, 41(3):1–52, 2009.

[3] Adithya Bhaskar, Tushar Tomar, Ashutosh Sathe, and Sunita Sarawagi. Benchmarking and improving text-to-sql generation under ambiguity. *arXiv preprint arXiv:2310.13659*, 2023.

[4] Grant Blank and Karsten Boye Rasmussen. The data documentation initiative: the value and significance of a worldwide standard. *Social Science Computer Review*, 22(3):307–318, 2004.

[5] Harrison Chase. LangChain, 10 2022. URL `https://github.com/langchain-ai/langchain`.

[6] Xu Chu, Ihab F Ilyas, Sanjay Krishnan, and Jiannan Wang. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 international conference on management of data*, pages 2201–2206, 2016.

[7] William AV Clark and Karen L Avery. The effects of data aggregation in statistical analysis. *Geographical Analysis*, 8(4):428–438, 1976.

[8] Michał Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. Frustratingly short attention spans in neural language modeling. *arXiv preprint arXiv:1702.04521*, 2017.

[9] Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Jinshu Lin, Dongfang Lou, et al. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306*, 2023.

[10] Bonnie Erickson and Terry Nosanchuk. *Understanding data*. McGraw-Hill Education (UK), 1992.

[11] Avrilia Floratou, Fotis Psallidas, Fuheng Zhao, Shaleen Deep, Gunther Hagleither, Wangda Tan, Joyce Cahoon, Rana Alotaibi, Jordan Henkel, Abhik Singla, Alex Van Grootel, Brandon Chow, Kai Deng, Katherine Lin, Marcos Campos, Venkatesh Emani, Vivek Pandit, Victor Shnayder, Wenjing Wang, and Carlo Curino. Nl2sql is a solved problem... not! In *Proceedings of the Conference on Innovative Data Research (CIDR)*, 2024.

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

[13] Mouzhi Ge and Markus Helfert. Areview of information quality research.

[14] Gunther Hagleitner. Gpt-4's sql mastery. `https://medium.com/querymind/gpt-4s-sql-mastery-2cd1f3dea543`, April 2023.

[15] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 5549–5581. PMLR, 2023.

[16] Wei Jeng, Daqing He, and Jung Sun Oh. Toward a conceptual framework for data sharing practices in social sciences: A profile approach. *Proceedings of the Association for Information Science and Technology*, 53(1):1–10, 2016.

[17] Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. Kaggledbqa: Realistic evaluation of text-to-sql parsers. *arXiv preprint arXiv:2106.11455*, 2021.

[18] Nancy L Leech and Anthony J Onwuegbuzie. An array of qualitative data analysis tools: A call for data analysis triangulation. *School psychology quarterly*, 22(4):557, 2007.

[19] Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S Yu. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability. *arXiv preprint arXiv:2303.13547*, 2023.

[20] Jerry Liu. LlamaIndex, 11 2022. URL `https://github.com/jerryjliu/llama_index`.

[21] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

[22] Stuart E Madnick, Richard Y Wang, Yang W Lee, and Hongwei Zhu. Overview and framework for data and information quality research. *Journal of data and information quality (JDIQ)*, 1(1): 1–22, 2009.

[23] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. Raha: A configuration-free error detection system. In *Proceedings of the 2019 International Conference on Management of Data*, pages 865–882, 2019.

[24] Kyle Mahowald, Anna A Ivanova, Idan A Blank, Nancy Kanwisher, Joshua B Tenenbaum, and Evelina Fedorenko. Dissociating language and thought in large language models: a cognitive perspective. *arXiv preprint arXiv:2301.06627*, 2023.

[25] Imene Mami and Zohra Bellahsene. A survey of view selection methods. *Acm Sigmod Record*, 41(1):20–29, 2012.

[26] Anders Giovanni Møller, Jacob Aarup Dalsgaard, Arianna Pera, and Luca Maria Aiello. Is a prompt and a few samples all you need? using gpt-4 for data augmentation in low-resource classification tasks. *arXiv preprint arXiv:2304.13861*, 2023.

[27] Erhard Rahm, Hong Hai Do, et al. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[28] Vijayshankar Raman and Joseph M Hellerstein. Potter's wheel: An interactive data cleaning system. In *VLDB*, volume 1, pages 381–390, 2001.

[29] Amit Rudra and Shastri L Nimmagadda. Roles of multidimensionality and granularity in warehousing australian resources data. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pages 216b–216b. IEEE, 2005.

[30] Amit Shukla, Prasad Deshpande, Jeffrey F Naughton, et al. Materialized view selection for multidimensional datasets. In *VLDB*, volume 98, pages 488–499, 1998.

[31] Mary Vardigan, Pascal Heus, and Wendy Thomas. Data documentation initiative: Toward a standard for the social sciences. *International Journal of Digital Curation*, 3(1), 2008.

[32] Enzo Veltri, Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. Data ambiguity profiling for the generation of training examples. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 450–463. IEEE, 2023.

[33] Bing Wang, Yan Gao, Zhoujun Li, and Jian-Guang Lou. Know what i don't know: Handling ambiguous and unanswerable questions for text-to-sql. *arXiv preprint arXiv:2212.08902*, 2022.

[34] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[35] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle 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.

[36] Tao Yu, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. *arXiv preprint arXiv:1909.05378*, 2019.

[37] Min Zhang and Juntao Li. A commentary of gpt-3 in mit technology review 2021. *Fundamental Research*, 1(6):831–833, 2021.

[38] Weixu Zhang, Yu Wang, and Ming Fan. Towards robustness of large language models on text-to-sql task: An adversarial and cross-domain investigation. In *International Conference on Artificial Neural Networks*, pages 181–192. Springer, 2023.

[39] Wenqi Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. Data-copilot: Bridging billions of data and humans with autonomous workflow. *arXiv preprint arXiv:2306.07209*, 2023.

[40] Yi Zhang, Jan Deriu, George Katsogiannis-Meimarakis, Catherine Kosten, Georgia Koutrika, and Kurt Stockinger. Sciencebenchmark: A complex real-world benchmark for evaluating natural language to sql systems. *arXiv preprint arXiv:2306.04743*, 2023.

[41] Yusen Zhang, Xiangyu Dong, Shuaichen Chang, Tao Yu, Peng Shi, and Rui Zhang. Did you ask a good question? a cross-domain question intention classification benchmark for text-to-sql. *arXiv preprint arXiv:2010.12634*, 2020.