AmbiGraph-Eval: Can Large Language Models Resolve Ambiguities in Graph Queries?

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have shown great potential in generating database queries from natural language input, particularly for interacting with complex graph databases. However, real-world queries often contain inherent ambiguities, and the interconnected nature of graph structures can amplify these challenges, leading to unintended query results. To assess LLMs' ability to handle such ambiguities, we categorize graph ambiguity into three types: **Attribute ambiguity**, **Relationship ambiguity**, and **Attribute-Relationship ambiguity**, each with **Same Entity** and **Cross Entity** subcategories. We present AmbiGraph-Eval, a benchmark featuring real-world ambiguous queries and expert-verified graph queries. Our evaluation of 16 LLMs reveals that even state-of-the-art models struggle with ambiguous graph queries, highlighting significant challenges in ambiguity resolution. However, strategies such as pinpointing ambiguity locations, providing external knowledge, and adding inference steps can notably enhance LLM performance.

1 Introduction

Semantic parsing (Fowler, 2010; Kamath and Das, 2018), the process of translating natural language into domain-specific languages (DSLs), is a vital technology that enables intuitive interactions with task-oriented dialogue systems. By allowing users to issue commands or queries in natural language, it bridges the gap between human communication and machine execution. However, natural language is inherently ambiguous (Shah and Jinwala, 2015), often leading to multiple interpretations of the same expression. In contrast, DSLs, such as SQL for relational databases (Qin et al., 2022) or Cypher for graph-based databases (Green et al., 2019), require precise and unambiguous statements. This fundamental mismatch introduces a significant challenge in semantic parsing: effectively handling ambiguity

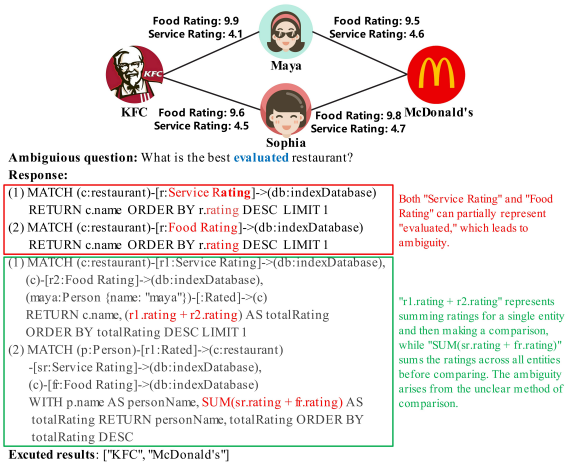

Figure 1: Ambiguous example in text-to-CQL task. Blue font denotes the problematic question span and red font means the "plausible" relationship.

to ensure accurate and intended translations from natural language to DSLs.

Recently, interest in addressing ambiguity in semantic parsing has increased (Wang et al., 2022; Bhaskar et al., 2023; He et al., 2024). These efforts mainly focus on addressing ambiguity in SQL semantic parsing tasks, particularly in tabular data formats such as relational databases or spreadsheets. However, these formats do not capture the complex interrelationships inherent in graph databases, which are widely used in numerous application scenarios, such as social media and payment platforms. Consequently, there has been limited exploration of ambiguity in the context of graph databases, where the rich and interconnected data structures present unique challenges for interpretation.

To be specific, natural language queries involving nodes and relationships in graphs often lead to multiple interpretations due to the structural diversity and richness of the data. For instance, as shown in Figure 1, the query "best evaluated restaurant" can produce different results depending on the criteria used for the rating. Since the graph lacks a

direct match for the relationship or attribute "evaluated", the system resorts to fuzzy matching with semantically similar attributes, such as "Service Rating" and "Food Rating", leading to the possible results highlighted in the red box. Moreover, the method of comparison is also ambiguous: the system could compare individual ratings (e.g., Maya's score) or aggregate ratings from all users, which leads to the optional results shown in the green box.

The presence of such ambiguities in interactive systems introduces significant risks. If semantic parsing techniques fail to detect and manage these ambiguities, the system may generate queries that deviate entirely from the user's intended request. This misalignment can lead to unnecessary data retrieval and computation, wasting valuable resources and time. In scenarios where real-time decision-making or resource efficiency is critical, these missteps can severely impact system performance, increasing operational costs and reducing overall effectiveness.

With the rapid advancement of large language models (LLMs) (Kojima et al., 2022), LLM-based semantic parsing has shown great potential in addressing complex and ambiguous user queries. These models leverage their linguistic knowledge to interpret nuanced inputs and confirm the intended query through interactive clarification with the user. However, despite these advances, current LLMs exhibit a notable limitation: *self-preference bias* (Ye et al., 2024). Since LLMs are trained using human feedback, the preferences of users providing feedback can influence the model's behavior. As a result, LLMs may prioritize one interpretation of a query while overlooking others, leading to misalignment between the system's output and the user's true intent, particularly in cases of multiple valid interpretations.

In this work, we evaluate the effectiveness of current LLMs and related techniques in handling ambiguity in graph queries. Specifically, we (1) systematically categorize types of ambiguity in graph queries, (2) introduce a curated dataset containing ambiguous user queries and corresponding graph data with human-labeled graph queries (Cypher), and (3) conduct a comprehensive evaluation of existing LLMs and relevant techniques to assess their ability to resolve these ambiguities effectively.

We introduce the concept of ambiguity in graph database queries, categorizing it into three primary types based on real-world data: **Attribute ambiguity**, arising from unclear attribute definitions; **Relationship ambiguity**, stemming from uncertainty in relationships or their attributes; and **Attribute-Relationship ambiguity**, where both attributes and relationships contribute to the uncertainty. Each type can be further divided into **Same Entity ambiguity**, where ambiguity pertains to a single entity, and **Cross Entity ambiguity**, where uncertainty arises from interactions across multiple entities.

To evaluate model performance, we propose *AmbiGraph-Eval*, a benchmark specifically designed for assessing the ability of LLMs to handle ambiguity in graph queries. AmbiGraph-Eval comprises 560 carefully curated ambiguous queries and corresponding graph database samples. Using this benchmark, we test 16 LLM models, analyzing their ability to resolve graph ambiguities, identifying model errors, and exploring strategies for improvement. Our findings indicate that reasoning capabilities offer limited benefits in this task. Instead, the models' understanding of graph ambiguity and their mastery of graph database query syntax play a more critical role. Through this study, we aim to advance the development of complex query generation and ambiguity resolution in graph query semantic parsing tasks. Our contributions can be summarized as follows:

- Based on real-world data observations, we introduced the concept of ambiguity in graph queries, categorizing it into three main types. This addresses a significant gap in the research on ambiguity in Text-to-Graph query tasks.
- We released AmbiGraph-Eval, a carefully curated dataset collected through a two-stage review process, designed to evaluate graph query ambiguity. Alongside this, we introduced a novel metric, i.e., *Ambiguity-Resolved Execution Accuracy (AREA)* metric, for measuring model performance in resolving graph query ambiguity.
- We evaluated 16 models using zero-shot, few-shot, chain-of-thought, and ambiguity highlighting strategies, providing a thorough analysis of model performance and in-depth error analysis to offer insights into handling ambiguity.

2 Related Work

Graph Query Parsing Benchmark. For the Graph query semantic parsing task, benchmarks like OVERNIGHT (Wang et al., 2015) and GrailQA (Gu et al., 2021) assess model generalization across various domains, aiding in the development of more adaptable KBQA (Knowledge Base

Question Answering) models. KQA Pro (Cao et al., 2020) and MetaQA (Zhang et al., 2018) focus on evaluating models' multi-hop reasoning capabilities in knowledge bases. However, while these benchmarks address complex question-answering tasks, they do not consider ambiguity, as each natural language query is paired with a single logical form. To address this gap, we propose AmbiGraph-Eval, a benchmark specifically designed to evaluate LLMs' ability to handle ambiguity in graph queries.

Ambiguity In NLP tasks. Ambiguity is a core aspect of human language, balancing conciseness with precision, but it presents significant challenges in interpretation. It has been widely studied in NLP tasks like syntactic parsing (Church and Patil, 1982; Koller et al., 2008) and coreference resolution (Poesio and Artstein, 2005). Recent work, such as Mu et al. (2023), addressed ambiguity in user queries for improved code generation, and Liu et al. (2023) highlighted how even advanced language models struggle with ambiguity. Despite the increased focus on ambiguity in NLP, its study in graph structures remains underexplored. Our work bridges this gap by categorizing and analyzing ambiguity within graph queries, offering insights into how LLMs handle ambiguity in graph-based data.

3 Ambiguity Categorization

In this section, we introduce our systematic categorization of ambiguity in graph query parsing, based on analysis of 22 graph databases. By examining node and edge attributes, along with data heterogeneity across sources, we identified three primary types of ambiguity: **Attribute ambiguity**, **Relationship ambiguity**, and **Attribute-Relationship ambiguity**. Each category is further divided into **Same Entity ambiguity** and **Cross Entity ambiguity**, ensuring comprehensive coverage of potential ambiguities in graph databases (Figure 2).

Attribute ambiguity occurs when a user queries an entity's attribute, but multiple plausible attributes in the graph could match the query. This often arises in graphs with different data sources or representations of similar information. For example, querying the citation count of a paper titled "CodeHalu" may yield multiple counts from sources like Google Scholar or Semantic Scholar, creating uncertainty about which count the user intends. Ambiguity can also occur in comparisons, such as "Which paper has the highest citation count?" if the citation source is not specified. We divide this ambiguity into two sub-types: **Same Entity ambiguity**: When a single entity has multiple attributes that could match the user's intent, creating uncertainty about the correct attribute; **Cross Entity ambiguity**: When comparing multiple entities without clear specification of which attribute should be used, making the comparison unclear.

Relationship ambiguity arises when a query involves an entity's relationships, but the specific relationship required to resolve the query is unclear. This may happen when multiple relationships could potentially satisfy the query, or when the relationships themselves are complex. For instance, asking for an architect's "contributions" could refer to 'design contributions' or 'project contributions', creating ambiguity in how the query should be answered. Similar uncertainty can arise in comparative queries, such as "Who is the best doctor according to patient reviews?" where the review criteria (e.g., score vs. rating) are unclear. Similarly, this ambiguity can be divided into two sub-types: **Same Entity ambiguity**: When a single entity has multiple relationships, leading to uncertainty about which relationship is relevant to the query; **Cross Entity ambiguity**: When comparing entities based on relationships, but the criteria or standards for comparison are unclear.

Attribute-Relationship ambiguity occurs when both an entity's attributes and relationships are involved, making it unclear whether the information should be derived from the entity's attributes or its relationships. For example, the query "What is the rating for the movie Inception?" could refer to an internal attribute (e.g., 'IMDb rating') or an external relationship (e.g., reviews from external sources). Similarly, in the query "Which city hall has the highest evaluation?", the ambiguity arises from whether the focus is on the internal rating or external reviews. **Same Entity ambiguity**: When the boundary between an entity's attributes and its relationships is unclear, causing uncertainty in how to answer the query. **Cross Entity ambiguity**: When comparing multiple entities where both attributes and relationships could influence the answer, it is unclear which to prioritize.

4 AmbiGraph-Eval Benchmark

AmbiGraph-Eval evaluates large language models (LLMs) on their ability to generate syntactically correct and semantically appropriate logical lan-

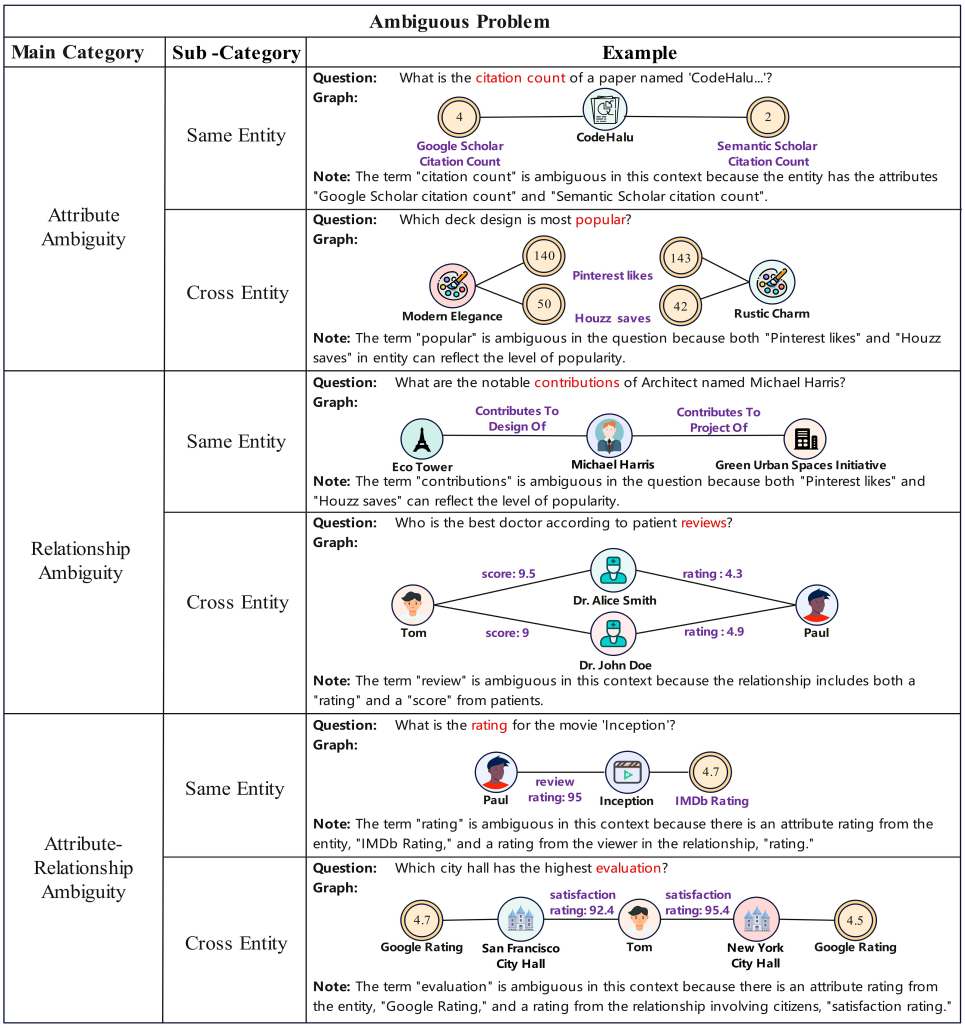

Figure 2: Ambiguous problem categories in Text-to-Graph query task. The red font with a dashed line denotes the ambiguous question span. The purple font means a related concept (Attributes or Relationships) to the red span. Note sentence explains why the example is ambiguous.

Category	Sub-Category	Samples	Ambig/Unambig
Attribute	same-entity	86	50/36
	cross-entity	101	65/36
Relationship	same-entity	86	49/37
	cross-entity	101	56/45
Attribute-Relationship	same-entity	94	57/37
	cross-entity	92	59/33

Table 1: Detailed statistics of categories, subcategories, and sample quantities in AmbiGraph-Eval.

guages in response to ambiguous user queries. The dataset consists of natural language queries with inherent ambiguities and corresponding graph data, where a single query may yield multiple valid logical language outputs, assessing the model's ability to resolve ambiguity.

The dataset is built in two stages: **data collection** and **human review**, as shown in Figure 3. During data collection, ambiguous graph data is sourced through real and synthetic data. In the human review stage, **experts refine and validate the data and corresponding answers** to ensure accuracy. Further details of this process are in Appendix A.1, and benchmark statistics are provided in Table 1.

5 Experiments

In this section, we benchmark various open-source and closed-source models using AmbiGraph-Eval, providing a comparative analysis of their performance. Cypher (Francis et al., 2018) is used as the validation language for all experiments.

5.1 LLM Models

To comprehensively analyze the different ambiguities of various competitive LLMs in AmbiGraph-Eval, we evaluate 7 closed-source LLMs, including O1-mini(OpenAI, 2024), GPT-4(OpenAI, 2023), GPT-4o(OpenAI, 2024), GPT-3.5(OpenAI,

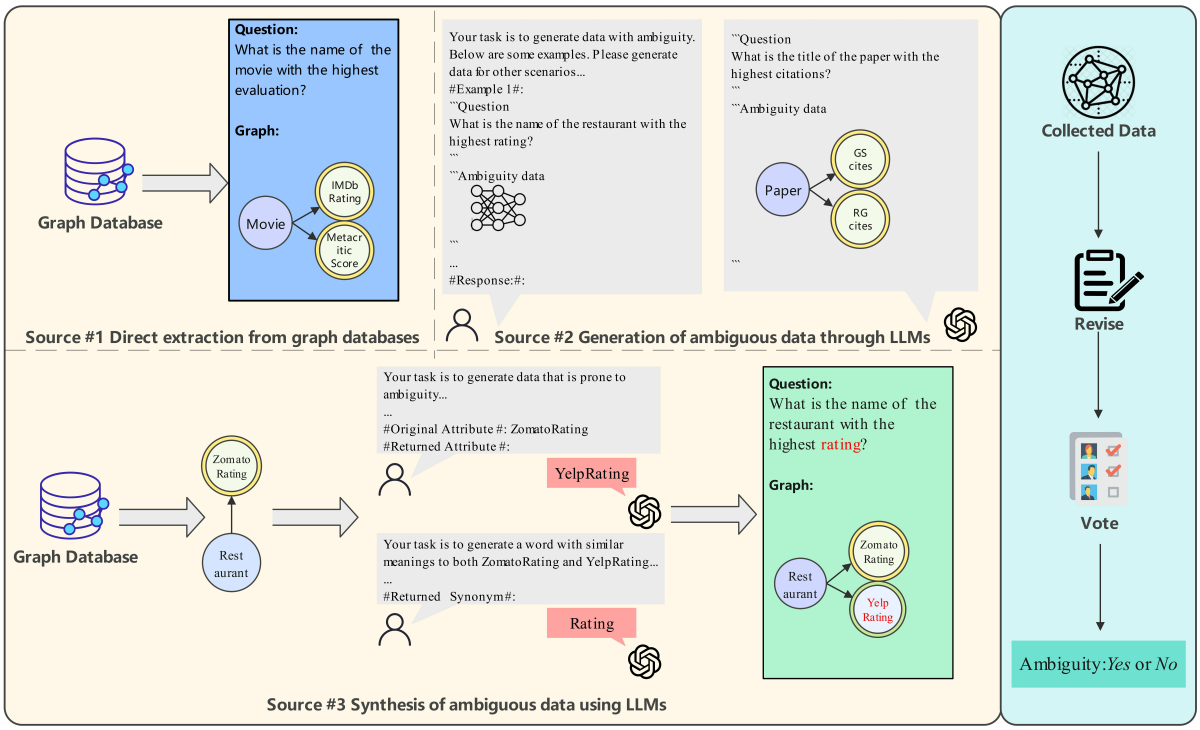

Figure 3: The construction process of AmbiGraph-Eval. it is divided into two stages: data collection and human review.

2022), Gemini-1.5-pro(Google, 2024), Claude-3.5-Sonnet(Anthropic, 2024), Qwen-Plus(Tongyi, 2024). We also evaluate 9 open-source LLMs, including Qwen-2.5 series(Tongyi, 2024), LLaMA-3.1 series(Meta, 2024), DeepSeek-V2.5(deepseek, 2024), Mistral-7B-V0.3(mistral, 2024). The evaluation is conducted by API calls or 4x NVIDIA A40 GPUs.

5.2 Metrics

In the text-to-CQL task (Guo et al., 2022), the goal is to generate CQL code that not only compiles successfully but also retrieves the correct results from the database. Traditional static metrics, such as syntax match, assess syntax and structure but do not capture execution performance. Code can score well on these metrics yet fail during compilation or execution. Therefore, we focus on metrics that verify whether the generated CQL code compiles and executes as expected.

Ambiguity in database queries can affect result accuracy. To evaluate how well models handle ambiguity, we introduce **Ambiguity-Resolved Execution Accuracy (AREA)**. When ambiguity is detected, the model must generate multiple Cypher queries, each retrieving a single correct result.

Let the target result be $R_{\text{target}} = \{t_1, t_2, \ldots, t_k\}$, and the model-generated results be $R(C_1), R(C_2), \ldots, R(C_n)$. Each query must return only one element from R_{target}. The validity of each query is determined by the function g:

$$g(R(C_i)) = \begin{cases} 1 & \text{if } |R(C_i)| = 1 \text{ and } R(C_i) \subseteq R_{\text{target}} \\ 0 & \text{otherwise} \end{cases}$$

A query is valid only if it returns a single result within R_{target}. A set coverage function h is introduced to check if the combined results cover the entire target set:

$$h\left(\{R(C_i)\}_{i=1}^{n}, R_{\text{target}}\right) = \begin{cases} 1 & \text{if } \bigcup_{i=1}^{n} R(C_i) = R_{\text{target}} \\ 0 & \text{otherwise} \end{cases}$$

The metric AREA for the task is defined as:

$$AREA = h(\{R(C_1), R(C_2), \ldots, R(C_n)\}, R_{\text{target}})$$

For the entire task set, the percentage of successful tasks is calculated as:

$$P_{\text{AREA}} = \frac{1}{N} \sum_{i=1}^{N} AREA_i \times 100\%$$

To our knowledge, AREA is the first metric that accurately reflects the ability of LLMs to handle ambiguity in graphs during text-to-CQL through actual execution tests.

Model	Attribute			Relationship			Attribute&Relationship			Average
	SE	CE	Avg.	SE	CE	Avg.	SE	CE	Avg.	
GPT-4o	75.58	**88.19**	**80.64**	65.12	**92.08**	80.13	53.19	**83.70**	**68.28**	**76.79**
LLaMA-3.1-Instruct-405B	79.07	78.22	78.61	77.91	**92.08**	**85.56**	**57.45**	71.74	64.51	76.25
O1-mini	**81.40**	78.22	79.78	74.42	83.17	79.14	54.26	68.48	61.29	73.39
Qwen-Plus	74.42	62.38	67.91	88.37	77.23	82.35	35.11	**83.70**	59.14	69.82
Qwen-2.5-Instruct-72B	75.58	57.43	65.78	89.53	78.22	83.42	32.98	78.26	55.38	68.21
Claude-3.5-Sonnet	72.09	69.31	70.59	68.60	77.23	73.36	**57.45**	42.39	50.00	64.64
GPT-4	55.81	67.33	62.03	**91.86**	79.21	85.03	35.11	45.65	40.32	62.50
DeepSeek-V2.5	55.81	60.40	58.29	81.40	72.28	76.47	42.55	51.09	50.00	61.61
Gemini-1.5-Pro	63.95	68.32	66.31	67.44	80.20	74.33	32.98	38.04	35.48	58.75
GPT-3.5	31.40	17.82	24.06	74.42	62.38	67.91	19.15	34.78	26.88	39.64
Qwen-2.5-Instruct-7B	48.84	25.74	36.36	66.28	31.68	48.13	28.72	30.43	29.57	38.04
LLaMA-3.1-Instruct-8B	44.19	26.73	34.76	58.14	18.81	36.90	31.91	21.74	26.88	32.86
Mistral-7B-Instruct-V0.3	27.90	9.90	18.18	44.19	10.89	26.20	18.09	15.22	16.67	20.36

Table 2: Evaluation results of 13 models on AmbiGraph-Eval. SE denotes same-entity. CE denotes cross-entity. Results are measured by P_{AREA} (%)

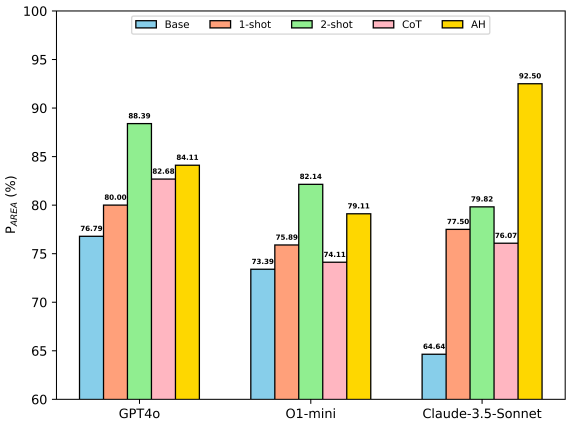

Figure 4: The performance of GPT-4o, O1-mini, and Claude-3.5-Sonnet on AmbiGraph-Eval across different strategies.

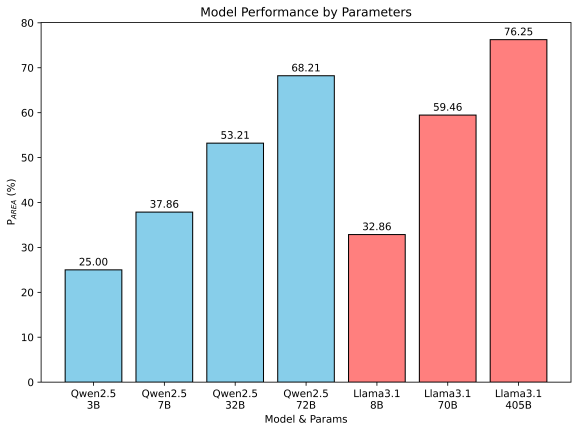

Figure 5: Scaling the number of models' parameters on AmbiGraph-Eval.

5.3 Results & Analysis

5.3.1 Zero-shot Results

In Table 2, we evaluate the zero-shot performance on our **AmbiGraph-Eval** Benchmark. The findings reveal substantial performance disparities among models in resolving graph data ambiguities. GPT-4o consistently demonstrated the best overall performance, excelling in both attribute and relationship ambiguity tasks. In contrast, while O1-mini delivered balanced results across most tasks, it was surpassed by GPT-4o in handling ambiguities requiring a deeper understanding of language and data structures. O1-mini, though adept at reasoning in tasks like coding or mathematical problem-solving, did not perform as well in this benchmark. Additionally, some models displayed task-specific strengths but struggled with distinguishing between same-entity and cross-entity scenarios.

Attribute ambiguity. O1-mini achieves the highest performance in same-entity (SE) attribute ambiguity tasks, with GPT-4o and LLaMA-3.1 closely following. These models show strong abilities in resolving ambiguities related to attributes within a single entity. However, in cross-entity (CE) attribute ambiguity tasks, GPT-4o significantly outperforms all models, showcasing superior reasoning capabilities in comparing attributes across different entities. LLaMA-3.1 and O1-mini also perform well on CE tasks, though not as effectively as GPT-4o. GPT-4o and O1-mini both maintain consistent performance across SE and CE tasks, while models like Qwen-2.5 show competency in SE tasks but struggle with CE tasks, suggesting difficulties in managing cross-entity ambiguity.

Relationship ambiguity. LLaMA-3.1 and GPT-4 both excel in handling relationship ambiguity tasks. Although GPT-4o shows the strongest over-

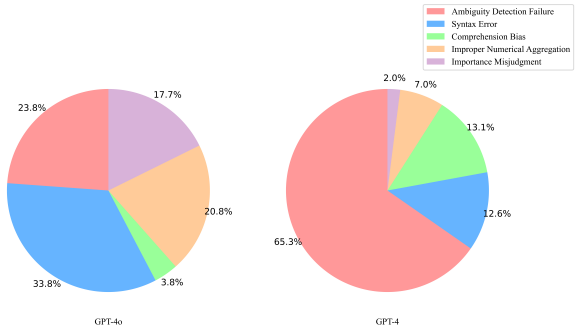

Figure 6: Error Type Distribution of GPT-4 and GPT-4o on AmbiGraph-Eval.

all performance, it shows relative weaknesses in addressing SE relationship ambiguity, indicating limitations in managing ambiguity within relationships involving the same entity. Conversely, in CE relationship ambiguity tasks, GPT-4o and LLaMA-3.1 achieve the highest scores, highlighting their strong abilities to handle CE ambiguity. These models particularly excel when comparing relationships across different entities, effectively tackling the complex challenges posed by such ambiguities.

Attribute-relationship ambiguity. LLaMA-3.1 and Claude-3.5 excel in SE attribute-relationship ambiguity tasks, demonstrating stability when resolving multi-dimensional ambiguities within the same entity. This reveals that these models face challenges in addressing attribute-relationship ambiguities within single entities. In CE tasks, GPT-4o continues to perform well, leading in cross-entity ambiguity challenges, with Qwen-Plus also showing notable strength. Overall, attribute-relationship ambiguity proves to be the most challenging category, with all models generally performing worse compared to handling attribute or relationship ambiguities independently. For instance, GPT-4 shows a drop in performance on these tasks compared to its handling of relationship ambiguities, exposing the difficulty of tackling both types of ambiguities simultaneously.

Scaling Performance. Figure 5 compares the performance of two open-source LLMs, Qwen-2.5 and LLaMA-3.1, across different parameter sizes. The results indicate that increasing model size can significantly improve performance, with LLaMA-3.1-405B achieving results comparable to GPT-4o.

5.3.2 In-Context Learning Strategies

Figure 4 illustrates the performance of GPT-4, O1-mini, and Claude-3.5-Sonnet on AmbiGraph-Eval using different strategies, including few-shot, Chain of Thought (CoT), and Ambiguity Highlight (AH), the latter of which explicitly identifies the ambiguity within the data.

Few-shot. The one-shot and two-shot strategies not only enhance the model's understanding of ambiguity but also aid in generating syntactically correct Cypher Query Language (CQL) code. The one-shot strategy provides a single example that demonstrates how to resolve ambiguity, offering a correct CQL query as a syntactic reference, thus reducing the likelihood of errors. However, its limitations become apparent in handling more complex queries. The two-shot strategy extends this advantage by presenting two distinct examples, allowing the model to grasp ambiguity from multiple perspectives and offering additional correct CQL references, thereby further improving syntactic accuracy. In summary, both strategies assist the model in better understanding ambiguity and generating accurate CQL code by providing illustrative examples.

Chain of Thought (CoT). In addressing complex graph ambiguities, CoT strategy first explicates the sources and nature of ambiguities, guiding the model through each potential ambiguity step by step. This approach enables a more thorough analysis of ambiguity, helping the model avoid prematurely transitioning to code generation. By fostering clear reasoning around ambiguities, the model can more precisely grasp task intricacies, resulting in more appropriate Cypher Query Language generation. Nevertheless, while CoT has shown strong performance in certain reasoning tasks, its effectiveness has not universally surpassed other methods. One possible explanation is that graph database ambiguities may sometimes be better resolved through concrete examples or explicit prompts rather than gradual reasoning processes. For ambiguity issues requiring intricate reasoning, CoT's emphasis on the continuity of reasoning chains may detract from addressing key aspects of certain ambiguous scenarios, potentially resulting in suboptimal CQL outputs. Thus, while the CoT strategy holds distinct advantages in fostering a deeper understanding of ambiguity prior to code generation, its effectiveness is modulated by the complexity of the reasoning involved and the diversity of data encountered. Of particular note is the relatively limited improvement observed in the O1-mini model, suggesting that for models with already robust reasoning capacities, the benefits of CoT may be marginal.

Ambiguity Highlighting (AH). The Ambigu-

ity Highlighting strategy aids models by explicitly identifying ambiguities in the data, alleviating the need for the model to reason through them. With this approach, the model no longer has to discern where the ambiguity lies, as these details are clearly marked in the input. Consequently, the model's primary task shifts to efficiently generating accurate queries based on the known ambiguities. The effectiveness of the AH strategy is particularly evident with the Claude-3.5, which demonstrates marked improvement when the ambiguities are made explicit. This suggests that when ambiguity is clearly defined, the model can generate logically coherent and syntactically correct code more effectively. By removing the reliance on ambiguity recognition, the model's overall performance sees significant enhancement, especially in complex ambiguous scenarios where it can focus solely on producing precise queries aligned with the task requirements.

In summary, the central challenge for models in this task is not rooted in their reasoning capabilities but in their ability to accurately identify and handle ambiguity while generating correct CQL code. The Chain of Thought strategy, though valuable for organizing reasoning steps, has not shown a distinct advantage in this context because the task relies more on the model's ability to comprehend ambiguity directly and generate appropriate query language. While CoT clarifies reasoning processes, it struggles to address the complexities of ambiguity in graph data. In contrast, the AH strategy significantly improves performance by explicitly indicating the ambiguous aspects of the data. This allows models to bypass the ambiguity reasoning process and focus directly on code generation, which leads to more accurate CQL outputs, especially in models that benefit from clear ambiguity definitions. Moreover, the few-shot strategies, particularly two-shot, have proven highly effective. These examples not only enhance the model's understanding of ambiguity but also serve as references for generating syntactically correct CQL code. By providing concrete examples, few-shot strategies help models better capture the nuances of ambiguity and generate well-structured, syntactically valid CQL queries.

5.3.3 Distribution of Errors

The figure 6 illustrates the distribution of five types of errors made by GPT-4 and GPT-4o in AmbiGraph-Eval: **Ambiguity Detection Failure, Syntax Error, Comprehension Bias, Improper Numerical Aggregation, and Importance Misjudgment**. These error types were identified through an analysis of the models' generated outputs. **Ambiguity Detection Failure** refers to the model's inability to identify ambiguity in the task accurately, **Syntax Error** indicates issues with the syntax of the generated Cypher Query Language (CQL), **Comprehension Bias** involves the model's misunderstanding of the task or data structure, **Improper Numerical Aggregation** relates to errors in aggregating numerical values, and **Importance Misjudgment** refers to the model's incorrect prioritization during comparisons. The results demonstrate that GPT-4 struggles with ambiguity detection, frequently missing key ambiguous information, while GPT-4o performs better in this area. However, GPT-4 exhibits a relatively lower proportion of other types of errors, which may be due to its failure to detect ambiguity, thereby avoiding further issues in tasks such as numerical aggregation and importance ranking. In contrast, GPT-4o, despite excelling in ambiguity detection, encounters more syntax errors when generating CQL. While GPT-4 performs better in syntax generation, its ability to handle complex tasks is hindered by improper ambiguity management. This also explains why the 2-shot strategy improves GPT-4o's performance more significantly compared to the Ambiguity Highlighting strategy. The 2-shot strategy, by providing examples with CQL syntax rules, helps GPT-4o achieve better results in syntax generation, while GPT-4o already performs well in ambiguity detection, rendering the AH strategy's benefits relatively limited. The 2-shot strategy effectively compensates for GPT-4o's syntactical weaknesses, leading to greater overall improvement.

6 Conclusion

In this work, we present **AmbiGraph-Eval**, a novel benchmark designed to evaluate LLMs' ability to handle ambiguity in graph database queries. We tested 16 popular LLMs, both proprietary and open-source, on this benchmark. The results highlight the challenge of resolving graph ambiguity, with many models struggling to consistently generate correct Cypher queries. Moreover, our analysis reveals that strong general reasoning abilities do not always correlate with improved performance in handling ambiguous graph data, emphasizing the importance of models' understanding of ambiguity and database query syntax.

Limitations

Our benchmark has several limitations that present opportunities for future improvement:

- **Atomic-level focus:** The ambiguity definitions and classifications in Text-to-Graph queries are primarily based on atomic graph components—attributes, nodes, and relationships. However, more complex scenarios, such as multi-hop reasoning and cross-graph interactions, are not yet considered. Future work will explore addressing these higher-order ambiguities.

- **Limited dataset diversity:** While AmbiGraph-Eval provides valuable insights into handling ambiguity in graph queries, the dataset may not fully capture the breadth and complexity of real-world graph databases. Domain-specific or rare ambiguities may be underrepresented, which could limit the generalizability of the benchmark.

- **Lack of experiments in other query languages and logical forms:** While our work primarily focuses on Cypher queries, the benchmark is structured to allow easy extension to other query languages and logical forms. However, we have not yet explored or validated its performance beyond Cypher. Future research could expand on this by conducting experiments across a variety of query languages and logical representations to assess the benchmark's broader applicability.

References

Anthropic. 2024. Claude 3.5 Sonnet.

Adithya Bhaskar, Tushar Tomar, Ashutosh Sathe, and Sunita Sarawagi. 2023. Benchmarking and improving text-to-sql generation under ambiguity. *ArXiv*, abs/2310.13659.

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2020. Kqa pro: A dataset with explicit compositional programs for complex question answering over knowledge base. *arXiv preprint arXiv:2007.03875*.

Kenneth Church and Ramesh Patil. 1982. Coping with syntactic ambiguity or how to put the block in the box on the table. *American Journal of Computational Linguistics*, 8(3-4):139–149.

deepseek. 2024. DeepSeek: Brand new experience, redefining possibilities.

Martin Fowler. 2010. *Domain-specific languages*. Pearson Education.

Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An evolving query language for property graphs. *Proceedings of the 2018 International Conference on Management of Data*.

Google. 2024. Our next-generation model: Gemini 1.5.

Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Martin Schuster, Petra Selmer, and Hannes Voigt. 2019. Updating graph databases with cypher. *Proceedings of the VLDB Endowment*, 12(12):2242–2254.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21, page 3477–3488, New York, NY, USA. Association for Computing Machinery.

Aibo Guo, Xinyi Li, Guanchen Xiao, Zhen Tan, and Xiang Zhao. 2022. Spcql: A semantic parsing dataset for converting natural language into cypher. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 3973–3977, New York, NY, USA. Association for Computing Machinery.

Xinyi He, Mengyu Zhou, Xinrun Xu, Xiaojun Ma, Rui Ding, Lun Du, Yan Gao, Ran Jia, Xu Chen, Shi Han, et al. 2024. Text2analysis: A benchmark of table question answering with advanced data analysis and unclear queries. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18206–18215.

Aishwarya Kamath and Rajarshi Das. 2018. A survey on semantic parsing. *arXiv preprint arXiv:1812.00978*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Alexander Koller, Michaela Regneri, and Stefan Thater. 2008. Regular tree grammars as a formalism for scope underspecification. In *Proceedings of ACL-08: HLT*, pages 218–226, Columbus, Ohio. Association for Computational Linguistics.

Alisa Liu, Zhaofeng Wu, Julian Michael, Alane Suhr, Peter West, Alexander Koller, Swabha Swayamdipta, Noah Smith, and Yejin Choi. 2023. We're afraid

language models aren't modeling ambiguity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 790–807, Singapore. Association for Computational Linguistics.

Meta. 2024. Introducing Llama 3.1: Our most capable models to date.

mistral. 2024. mistral-7b.

Fangwen Mu, Lin Shi, Song Wang, Zhuohao Yu, Binquan Zhang, ChenXue Wang, Shichao Liu, and Qing Wang. 2023. Clarifygpt: Empowering llm-based code generation with intention clarification. *ArXiv*, abs/2310.10996.

OpenAI. 2022. Introducing ChatGPT.

OpenAI. 2023. GPT-4 technical report.

OpenAI. 2024. GPT-4o contributions.

OpenAI. 2024. OpenAI o1-mini: Advancing cost-efficient reasoning.

Massimo Poesio and Ron Artstein. 2005. The reliability of anaphoric annotation, reconsidered: Taking ambiguity into account. In *Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky*, pages 76–83, Ann Arbor, Michigan. Association for Computational Linguistics.

Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, et al. 2022. A survey on text-to-sql parsing: Concepts, methods, and future directions. *arXiv preprint arXiv:2208.13629*.

Unnati S Shah and Devesh C Jinwala. 2015. Resolving ambiguities in natural language software requirements: a comprehensive survey. *ACM SIGSOFT Software Engineering Notes*, 40(5):1–7.

Tongyi. 2024. Qwen2.5: A Party of Foundation Models!

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

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, Beijing, China. Association for Computational Linguistics.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, et al. 2024. Justice or prejudice? quantifying biases in llm-as-a-judge. *arXiv preprint arXiv:2410.02736*.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

A Appendix

We provide some extra information about our benchmark as supplementary materials. The appendix is organized into two sections:

- The process of data collection;
- Case Study.

A.1 The Process of data collection

A.1.1 Data Collection

The data collection process is divided into three complementary approaches, each aimed at capturing different types of ambiguity and expanding the coverage of the dataset. These approaches are: **direct extraction of ambiguous graph data from existing databases**, **synthesis of ambiguous data from unambiguous graph data using LLMs**, and **full generation of additional ambiguous data through LLMs to broaden the dataset's scope**. The combination of these approaches ensures that the dataset captures both naturally occurring and artificially generated ambiguities, enhancing its robustness and generality. The data synthesis and generation process is fully automated, utilizing the capabilities of GPT-4 and GPT-4o to handle this task.

Direct extraction from graph databases. This approach focuses on leveraging existing graph databases that inherently contain ambiguous data. During the classification phase, graph data that presents ambiguity—either through its structure or through conflicting or incomplete information—is extracted and standardized. These extracted data segments may represent incomplete graph structures or specific entities and relationships from larger graphs. Since these data do not inherently come with ambiguous questions or answers, we manually create such questions by identifying potential sources of ambiguity within the graph (e.g., overlapping attributes, unclear relationships). We also annotate each entry to highlight where and why ambiguity arises, providing valuable context for both model evaluation and dataset users. This process ensures that the dataset includes real-world

examples of ambiguous graph data, laying the foundation for more comprehensive evaluations.

Synthesis of ambiguous data using LLMs. In order to expand the range of ambiguities represented in the dataset, we apply LLMs to unambiguous graph data, transforming it into ambiguous scenarios. This method is inspired by prior work on generating synthetic data to model real-world ambiguities. The synthesis process focuses on three key types of ambiguity: **attribute ambiguity**, **relationship ambiguity**, and **attribute-relationship ambiguity**. For **attribute ambiguity**, the process begins by identifying an entity x and its attribute y from the unambiguous graph data. We then prompt the LLM to generate a synonym for y, denoted as z, which could plausibly be confused with the original attribute. This transformation allows us to create ambiguous questions Q where the distinction between y and z becomes unclear. The corresponding answers are defined as the specific values of both y and z, forcing the model to disambiguate based on subtle differences. For **relationship ambiguity**, we first select a relationship r that links two entities, x and y. The LLM is then prompted to generate a semantically similar relationship r', along with a new entity z that could potentially be linked by r'. This newly generated graph structure introduces ambiguity in interpreting which relationship applies in a given query, and we generate questions Q that explore these ambiguities. The answers depend on correctly identifying the appropriate relationship and corresponding attributes of y or z. In the case of **attribute-relationship ambiguity**, both an attribute and a relationship are involved in the ambiguity. We begin by selecting an entity x with a specific relationship r. The LLM generates a synonym attribute y for x and an alternative relationship r' that could confuse the query's target. Questions Q are then created that ambiguously reference either the attribute or the relationship, with answers requiring the model to distinguish between them. This synthesis process allows us to systematically introduce a wide range of ambiguities, ensuring the dataset reflects various real-world disambiguation scenarios. A detailed description of the instructions for generating data with attribute ambiguity is provided in Table 3.

Generation of ambiguous data through LLMs. While the first two methods focus on either extracting real-world ambiguities or synthesizing them from existing data, this approach generates new ambiguous data entirely through LLMs. We use previously validated real-world and synthesized examples as seed data, prompting LLMs to generate new ambiguous questions and answers by altering or extending the seed examples into different contexts. This step is critical for diversifying the dataset, ensuring that it not only covers standard types of ambiguities but also explores novel ambiguity cases that may arise in practice. Additionally, this approach allows us to test LLMs' generalization capabilities by evaluating their performance on ambiguity types not directly mirrored in the seed data. Detailed instructions are provided in Table 4.

A.1.2 Human Review

To ensure that the collected data is both accurate and representative—meaning the questions genuinely contain ambiguity, the graph data structure is correct, the format is suitable for database storage, and the query results are reasonable—we implemented a rigorous, multi-level review process. Team members meticulously examined each type of ambiguous data to ensure it met the expected criteria for ambiguity while minimizing potential biases. This process covered both data generated by LLMs and ambiguous questions crafted by humans, preventing the introduction of bias from either source. Following the initial review, team members voted on the revised data. If any member deemed a particular data point unreasonable, it was removed. Through this stringent filtering process, we ultimately compiled 560 validated ambiguous data points. This review mechanism ensures the high quality and fairness of the dataset, providing a solid foundation for subsequent research.

A.2 Case Study

Table 5-9 provide case studies of GPT-4 in various types of error.

#First Step: Generate an ambiguous attribute

Instruction:
Your task is to generate data that is prone to ambiguity. For example, when I give you 'ZomatoRating', you need to return a property similar to 'YelpRating', so that when I want a Rating, both of these Ratings are reasonable. Please generate using the following format:
Given Attributes:
```\n ZomatoRating \n```
Returned Attributes:
```attribute\n YelpRating \n```
Below is my input:
Original Attributes:
```\n Google Scholar citation count \n```
Returned Attributes:

Response:
```attribute\n Ressearch Gate citation count \n```

#Second Step: Generate a synonym for ambiguous attributes

Instruction:
Your task is to generate a word with similar meanings to both 'Google Scholar citation count' and 'Ressearch Gate citation count'. Do not return the same words as given word. Please generate using the following format:
Given Words:
Word A: ZomatoRating
Word B: YelpRating
Returned Synonym:
```synonym\n Rating \n```
Below is my input:
Given Words:
Word A: Google Scholar citation count
Word B: Ressearch Gate citation count
Returned Synonym:

Response:
```synonym\n citation count \n```

Table 3: A detailed instruction on ambiguous data synthesis.

Instruction:
Your task is to generate data with ambiguity. Here are some examples; please generate data for other scenarios such as according to the format in the examples.
Example #1:
```` Ambiguity question \nWhat is the patient capacity for the hospital named 'Green Valley Medical Center'? \n```"
```Data

```
"entity_1": {
    "labels": "hospital",
    "props": {
        "name": "Green Valley Medical Center",
        "location": "New York, USA",
        "yearEstablished": 1998,
        "bedCapacity": 300,
        "emergencyCapacity": 50
    }
}
                ...
```
```Structure\n None \n```
```Ambiguity attributes\n ["bedCapacity","emergencyCapacity"] \n```
```Ground Truth\n["300","50"] \n```
```Explanation \n The hospital in question is 'Green Valley Medical Center'. Due to the presence of two attributes related to patient capacity, 'bed capacity' and 'emergency capacity', there is ambiguity in the data. The answers are 300 for bed capacity and 50 for emergency capacity respectively. \n```
Example #2:
...
Response:

**Response:**
```Ambiguity question\n What is the annual budget of the city hall located in 'Springfield'? \n ```"
```Data

```
"entity_1": {
    "labels": "cityHall",
    "props": {
        "name": "Springfield City Hall",
        "location": "Springfield",
        "yearestablished": 1890,
        "annualbudget": "5 million USD",
        "reportedbudget": "4.5 million USD"
    }
}
                ...
```
```Structure\n None \n```
```Ambiguity attributes\n ["annualbudget","reportedbudget"] \n```
```Ground Truth\n["5 million USD","4.5 million USD"] \n```
```Explanation \n The city hall in Springfield is associated with two different budget figures: '5 million USD' as the annual budget and '4.5 million USD' as the reported budget in a government report, creating ambiguity in the data. \n```

Table 4: A detailed instruction on ambiguous data generation.

**Ambiguity Detection Failure**

**Question:** What is the patient capacity for the hospital named 'Green Valley Medical Center'?

**Data:**

```
"entity_1": {
    "labels": "hospital",
    "props": {
        "name": "Green Valley Medical Center",
        "location": "New York, USA",
        "yearEstablished": 1998,
        "bedCapacity": 300,
        "emergencyCapacity": 50
    }
}
                ...
```

Structure: None

Generated Code

```Cypher
MATCH (h:hospital {name: 'Green Valley Medical Center'})
RETURN h.bedCapacity AS Patient_Capacity
```

Excepted Output	**Executed Result**
["300", "50"]	["300"]

Table 5: A detailed case study on Ambiguity Detection Failure by GPT-4.

Syntax Error

Question: Which arena hosted the most attendees last month?

Data:

```
"entity_1": {
    "labels": "arena",
    "props": {
        "name": "Madison Square Garden",
        "location": "New York",
        "ticketmasterAttendees": 150000,
        "eventbriteAttendees": 145000
    }
},
"entity_2": {
    "labels": "arena",
    "props": {
        "name": "Staples Center",
        "location": "Los Angeles",
        "ticketmasterAttendees": 140000,
        "eventbriteAttendees": 155000
    }
},
                ...
```

Structure:

```
[
    "entity_1",
    "rel_1",
    "entity_3"
],
                ...
```

Generated Code

```Cypher
MATCH (a:arena)-[:reportedAttendance]-(tp:ticketingPlatform {name: 'Ticketmaster'})
RETURN a.name AS Arena_Name\nORDER BY ticketmasterAttendees DESC\nLIMIT 1
```

```Cypher
MATCH (a:arena)-[:reportedAttendance]-(tp:ticketingPlatform {name: 'Eventbrite'})
RETURN a.name AS Arena_Name\nORDER BY eventbriteAttendees DESC\nLIMIT 1
```

Excepted Output	**Executed Result**
["Madison Square Garden","Staples Center"]	None

Table 6: A detailed case study on Syntex Error by GPT-4.

Improper Numerical Aggregation

Question: Which wetland area has the highest biodiversity index?

Data:

```
"entity_1": {
    "labels": "wetland",
    "props": {
        "name": "Everglades",
        "location": "Florida, USA",
        "birdlifeInternationalIndex": 85,
        "wwfIndex": 78
    }
},
"entity_2": {
    "labels": "wetland",
    "props": {
        "name": "Pantanal",
        "location": "Brazil",
        "birdlifeInternationalIndex": 80,
        "wwfIndex": 82
    }
}
                ...
```

Structure:

```
[
    "entity_1",
    "rel_1",
    "entity_3"
],
                ...
```

Generated Code

```Cypher
MATCH (w:wetland)\nRETURN w.name AS Wetland_Area, (w.birdlifeInternationalIndex
+ w.wwfIndex) AS Biodiversity_Index ORDER BY Biodiversity_Index DESC\nLIMIT 1
```

Excepted Output	**Executed Result**
["Everglades","Pantanal"]	["Everglades"]

Table 7: A detailed case study on Improper Numerical Aggregation by GPT-4.

Importance Misjudgment

Question: Which monument receives the most visitors annually?

Data:

```
"entity_1": {
    "labels": "monument",
    "props": {
        "name": "Eiffel Tower",
        "location": "Paris, France",
        "unescoWorldHeritageSite": true,
        "2019VisitorCount": 7000000,
        "2020VisitorCount": 2500000
    }
},
"entity_2": {
    "labels": "monument",
    "props": {
        "name": "Statue of Liberty",
        "location": "New York, USA",
        "unescoWorldHeritageSite": true,
        "2019VisitorCount": 4300000,
        "2020VisitorCount": 2600000
    }
},
                ...
```

Structure:

```
[
    "entity_1",
    "rel_1",
    "entity_3"
],
                ...
```

Generated Code

```Cypher
MATCH (m:monument) WITH m
ORDER BY m.2019VisitorCount, m.2020VisitorCount DESC LIMIT 1
```

Excepted Output	**Executed Result**
["Eiffel Tower","Statue of Liberty"]	["Statue of Liberty"]

Table 8: A detailed case study on Importance Misjudgment by GPT-4.