Evaluating and Enhancing LLMs for Multi-turn Text-to-SQL with Multiple Question Types

Ziming Guo[1], Chao Ma[1*], Yinggang Sun[2], Tiancheng Zhao[1], Guangyao Wang[1], Hai Huang[1]

[1]Harbin University of Science and Technology, Harbin, 150040, China.
[2]Harbin Institute of Technology, Harbin, 150001, China.

*Corresponding author(s). E-mail(s): machao8396@163.com;
Contributing authors: 2204050108@stu.hrbust.edu.cn;
23b903085@stu.hit.edu.cn; 1783467143@qq.com;
2204050124@stu.hrbust.edu.cn; hust_hh@vip.163.com;

Abstract

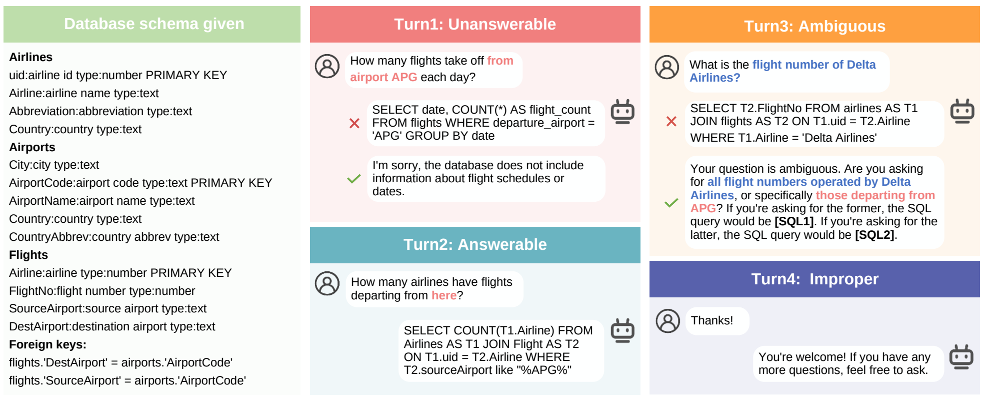
Recent advancements in large language models (LLMs) have significantly improved text-to-SQL systems. However, most LLM-based methods tend to focus narrowly on SQL generation, often neglecting the complexities inherent in real-world conversational queries. This oversight can result in unreliable responses, particularly for ambiguous questions that cannot be directly addressed with SQL. To address this gap, we propose MMSQL, a comprehensive test suite designed to evaluate the question classification and SQL generation capabilities of LLMs by simulating real-world scenarios with diverse question types and multi-turn Q&A interactions. Utilizing MMSQL, we assessed the performance of popular LLMs, including both open-source and closed-source models, and identified key factors influencing their performance in these contexts. Furthermore, we introduce an LLM-based multi-agent framework that employs specialized agents to identify question types and determine appropriate answering strategies. Our experiments demonstrate that this approach significantly enhances the model's ability to navigate the complexities of conversational dynamics, effectively handling user queries' diverse and intricate nature.

Keywords: Text-to-SQL, Dialogue Systems, Large Language Model, Intent Recognition, Multi-Agent

1 Introduction

Text-to-SQL bridges natural language and SQL queries, empowering non-technical users to access and analyze data without mastering complex SQL knowledge. The advent of LLMs, with their remarkable capacity for following instructions, has transformed the text-to-SQL domain. LLM-based methods have delivered remarkable outcomes across various text-to-SQL tasks [1, 2], while the assessment of their robustness is increasingly drawing attention. Current research often assumes that user queries are unambiguous, focusing on expanding the scope of databases and increasing the complexity of questions to elevate the difficulty of question-to-SQL mapping, thereby pushing the boundaries of text-to-SQL systems. [3, 4, 5, 6]. However, in practical scenarios, user queries are inherently dynamic and uncertain [7]. Assuming that all user queries can be answered may lead to the generation of hallucinated content and unreliable predictions, thereby undermining the dependability of these systems [8, 9].

Observing real-world conversations, it is common for users to engage in multi-turn dialogues with interrelated questions that do not readily translate into SQL. In such scenarios, relying on single-turn or assumptive responses can lead to suboptimal performance, as it may not fully address the user's needs or context. [7, 10, 11]. Figure 1 illustrates examples of responses to three common types of such questions: (1) *Unanswerable:* If the database lacks data, such as the flight departure times from an airport queried in Turn 1, the system needs to explain its inability to provide an answer. (2) *Ambiguous:* In Turn 3, the query about the flight number of Delta Airlines might refer to conditions from Turn 1 and 2, "from airport APG," or it might simply mean all flight numbers operated by Delta Airlines. The system needs to recognize this ambiguity and respond accordingly to avoid providing incorrect SQL responses. (3) *Improper:* Questions unrelated to the database content, such as casual conversation in Turn 4, should not prompt SQL-based responses. These examples highlight the necessity for reliable models capable of navigating various types of complex conversational dynamics. There is a need for more robust models that can effectively manage such challenges.



Airlines
uid:airline id type:number PRIMARY KEY
Airline:airline name type:text
Abbreviation:abbreviation type:text
Country:country type:text
Airports
City:city type:text
AirportCode:airport code type:text PRIMARY KEY
AirportName:airport name type:text
Country:country type:text
CountryAbbrev:country abbrev type:text
Flights
Airline:airline type:number PRIMARY KEY
FlightNo:flight number type:number
SourceAirport:source airport type:text
DestAirport:destination airport type:text
Foreign keys:
flights.'DestAirport' = airports.'AirportCode'
flights.'SourceAirport' = airports.'AirportCode'

How many flights take off from airport APG each day?

✗
```
SELECT date, COUNT(*) AS flight_count FROM flights WHERE departure_airport = 'APG' GROUP BY date
```

✓ I'm sorry, the database does not include information about flight schedules or dates.

How many airlines have flights departing from here?

```
SELECT COUNT(T1.Airline) FROM Airlines AS T1 JOIN Flight AS T2 ON T1.uid = T2.Airline WHERE T2.sourceAirport like "%APG%"
```

What is the flight number of Delta Airlines?

✗
```
SELECT T2.FlightNo FROM airlines AS T1 JOIN flights AS T2 ON T1.uid = T2.Airline WHERE T1.Airline = 'Delta Airlines'
```

✓ Your question is ambiguous. Are you asking for all flight numbers operated by Delta Airlines, or specifically those departing from APG? If you're asking for the former, the SQL query would be **[SQL1]**. If you're asking for the latter, the SQL query would be **[SQL2]**.

Thanks!

You're welcome! If you have any more questions, feel free to ask.

Fig. 1 A four-turn dialogue example that includes four different types of Q&A pairs.

Existing studies have only sporadically addressed the effective assessment and handling of multi-type questions. Most text-to-SQL research focuses on achieving high accuracy for single-type or single-round user questions [4, 6, 12, 13], often overlooking the need to develop systems capable of multi-turn dialogues that handle a variety of question types [3]. This limited scope can result in incorrect responses, especially for those questions that do not necessitate SQL solutions. It has been observed that even the most sophisticated LLMs struggle to effectively address these challenges [9]. Although recent studies have acknowledged the challenges of ambiguous and unanswerable questions [7]. However, existing solutions often resort to simple abstention when faced with vague or conversational elements, lacking granular designs to manage various question types, leading to passive responses and hindering practical applicability and robustness [10, 14, 15]. In summary, while individual studies have made some progress in evaluating and enhancing the reliability of text-to-SQL systems, these efforts fall short of a comprehensive solution to the problem.

In response to these challenges, we introduce the Multi-type and Multi-turn text-to-SQL test suite (MMSQL), a comprehensive benchmark engineered to evaluate the proficiency of LLMs in handling multi-turn text-to-SQL tasks across diverse question types. Utilizing MMSQL, we have undertaken an exhaustive performance analysis of leading LLMs, encompassing open-source and closed-source models. This analysis provides detailed insights, highlighting the promise of open-source models. It also reveals the suboptimal performance of models on specific question types and the varying effectiveness of strategies designed to resolve ambiguity. We propose an innovative LLM-based multi-agent framework, inspired by top-performing models and informed by these insights. This framework is anchored by a core Question Detector and Question Decomposer, tasked with identifying question types and determining appropriate answering strategies. It could provide potential SQL queries for ambiguous questions, accounting for multiple interpretations. Furthermore, the framework includes two supportive agents: the Schema Selector, which identifies and provides the essential subset of a database schema, and the SQL Refiner, which is dedicated to refining SQL queries. Our experiments, conducted on the MMSQL benchmark, have shown that this framework yields a significant performance improvement.

Our main contributions are summarized as follows:

1. We developed MMSQL, a comprehensive test suite designed to evaluate the performance of LLMs in multi-turn text-to-SQL conversations across diverse question types.
2. We provide an in-depth analysis of popular LLMs' performance on MMSQL, offering insights into their capabilities and identifying key factors that drive performance differences when handling a variety of question types.
3. We propose a novel multi-agent framework adept at addressing text-to-SQL tasks for various question types, evaluated on MMSQL, and demonstrated to be effective through a series of ablation experiments.

2 Related Work

2.1 Text-to-SQL

Text-to-SQL involves transforming natural language queries into SQL, representing a crucial area of natural language processing with many practical applications. Essential datasets include single-turn datasets such as WikiSQL [16], Spider, and BIRD [3], as well as multi-turn datasets such as CHASE [13], SParC [12], and CoSQL [17]. These datasets serve as benchmarks for evaluating the performance of text-to-SQL systems. In recent years, the emergence of pre-trained language models (PLMs) has driven significant progress in this domain. Models like SQLova [18], PICARD [19], and RAT-SQL [20] have achieved impressive results. However, despite their effectiveness, these specialized models require considerable resources and time for training. LLMs have revolutionized numerous NLP tasks, including text-to-SQL, surpassing previous models and establishing a new paradigm [21, 22]. Recent studies have focused on enhancing prompt design and developing complex, multi-stage frameworks that leverage LLMs to enhance performance [23]. Notably, LLM-based frameworks like DIN-SQL [24], which utilize chain-of-thought reasoning [25], decompose the problem into simpler sub-problems. MAC-SQL [6] and CHASE-SQL [5] employ multi-agent collaborative frameworks, demonstrating significant advancements in this field.

Recent studies highlight the inherent diversity and ambiguity of natural language, noting that not all user queries can be effectively translated into accurate SQL as the answer in practical applications [7, 10]. Datasets such as TriageSQL [14], NoisySP [7], and CoSQL have incorporated question type detection as part of the models' tasks. Previous work, TrustSQL [10] demands that models opt for abstention when faced with diverse unanswerable questions. AMBROSIA [9] and [26] propose testing text-to-SQL systems by parsing ambiguous questions into multiple potential SQL queries, thereby enhancing model usability in real-world scenarios. [26] improves beam search on T5 [27] to provide possible SQL responses for ambiguous questions and evaluate their effectiveness. Additionally, [15] employs questions enhanced dialogue augmentation to train open-source LLMs to identify question types. These studies focus solely on the model's ability to recognize question types or address specific problems, which may result in unhelpful abstentions or hallucinations due to specialized question-and-answer tailoring. Moreover, most work is limited to single-turn settings, whereas real interactions often involve ambiguity that must be addressed in multi-turn contexts. These limitations motivate our work, which evaluates and enhances LLMs for multi-turn dialogues with multiple question types.

2.2 LLM-based Agents

LLM-based agents have long captured the attention of researchers in both academia and industry [28]. With the expansion of web knowledge, LLMs are increasingly demonstrating intelligence levels comparable to humans. This evolution has sparked a growing interest in creating autonomous agents driven by LLMs. AutoGPT [29] and MetaGPT [30] enhance AI models by integrating a range of useful tools, allowing developers to create adaptable, conversational agents that can function in various modes by leveraging LLMs, human input, and other tools to complete tasks. In the realm of text-to-SQL parsing, frameworks such as MAC-SQL [6], MAG-SQL [31], and CHASE-SQL [5] employ multiple LLM-based agents to interpret SQL queries collaboratively. This can typically be summarized as a three-step process: retrieving relevant values, generating SQL, and refining the SQL output. These multi-agent systems have achieved leading results on the BIRD benchmark [3]. Drawing inspiration from these advancements, we developed a multi-agent framework incorporating a Question Detector Agent, designed to effectively manage the complexity and variety of user queries encountered in real-world applications.

3 MMSQL: Multi-type and Multi-turn Text-to-SQL Test Suite

3.1 Task Formulation

MMSQL aims to assess the effectiveness of language models in managing multi-type and multi-turn text-to-SQL tasks. By analyzing real-world text-to-SQL interactions, we have categorized four types of question-answer scenarios, as shown in Figure 1.

1. **Unanswerable**: These questions pertain to information that is not present in the database or exceeds the system's operational scope, such as when they involve external knowledge or web search capabilities beyond the SQL system. The system should acknowledge its inability to answer the question and explain the reason to the user. For example, a question about the airport's flight schedule each day, which is not present in the database, should be identified by the system as unanswerable.
2. **Answerable**: These questions can be directly answered using SQL queries based on the available database information. The system must generate an accurate SQL query reflecting the user's request and execute it to retrieve the required data. For example, in Turn 2, when asked, "How many airlines have flights departing from here?" the system correctly generates the SQL query by interpreting "here" within the given context to find airlines with flights departing from the specified location.
3. **Ambiguous**: These questions include terms that might correspond to multiple columns or values in the database, causing potential ambiguity in SQL query generation. In a multi-turn context, the system should identify and address this ambiguity by seeking clarification from the user. For example, in Turn 3, when asked, "What is the flight number of Delta Airlines?" the system should recognize the potential ambiguity—whether the user is asking for all flights or specific ones from a location like APG. It should ask for clarification (e.g., "Are you asking about all flights or specific ones?") and simultaneously propose possible SQL queries based on these interpretations. This approach ensures the system can adapt and refine its responses through ongoing dialogue.
4. **Improper**: These questions are irrelevant to the database or pertain to everyday conversation that does not require an SQL response. The system should recognize these questions and respond appropriately, without attempting to generate an

SQL query. For example, a casual "thank you" from the user should be met with a simple "You're welcome!" rather than an SQL statement.

In this task, the model is required to perform two key functions: first, to accurately determine the type of each question; and second, to attempt resolution and provide answers for questions categorized as ambiguous and answerable. This dual focus ensures that the model classifies questions effectively and actively generates appropriate SQL queries for complex scenarios.

3.2 Construction

The MMSQL data originates from two sources: samples refined from CoSQL and new samples generated by QDA-SQL [15]. QDA-SQL employs CoT to guide the generation of new samples, helping the LLM produce multi-turn Q&A pairs by reasoning step-by-step and constructing intermediate thoughts leading to the final answer. This uses iterative processes, random combinations of context relationships, and question types, followed by a refinement process to guide Gemini Pro in generating diverse multi-turn, multi-type datasets, ensuring each sample adheres to our defined question types. The samples generated by QDA-SQL demonstrated higher natural language annotation quality, with a 62% win rate, and included more complex text-to-SQL examples compared to the original dataset. We screened these initial samples to ensure they met high standards of question difficulty, classification accuracy, and diversity, resulting in a high-quality dataset. Necessary modifications and optimizations were made to the original datasets, transforming some instances into a curated MMSQL test set. Unanswerable and ambiguous questions are sometimes not easily distinguishable, so we ensured these characteristics were prominent enough during the manual annotation of the test set. In total, the dataset comprises 6,493 training rounds and 149 test rounds of dialogue.

3.3 Dataset Analysis

Table 1 summarizes the statistics of MMSQL, compared to two other multi-turn or multi-type datasets. A distinctive feature of MMSQL is that it includes four types of questions. MMSQL offers a significantly larger number of dialogues and features a higher average number of turns per dialogue, providing abundant multi-turn training data that covers a wide range of query types.

3.4 Evaluation Metrics

Following BIRD [3] and CoSQL [17], our evaluation metrics include Exact Matching (EM) and Execution Accuracy (EX). EM compares each predicted clause to the reference query, considering it is correct only if all components match, excluding values. EX evaluates the proportion of SQL where the execution results of both predicted and ground-truth SQL are identical. Interaction Execution Accuracy (IEX) is achieved when all SQL queries in a multi-turn interaction execute correctly. We also developed the Dual Assessment of Question Type Detection and Execution Accuracy (TDEX) to assess the comprehensive capability of text-to-SQL models with complex queries.

Table 1 Comparison of multi-turn or multi-type text-to-SQL datasets.

	SParC	CoSQL	NoisySP	AmbiQT	AMBROSIA	MMSQL
# Dialogues	4,298	3,007	-	-	-	**6493**
Total # turns	12,726	15,433	15,598	3,046	4,242	**38,666**
Avg. # Q turns	3.0	5.2	1	1	1	**6.0**
Avg. Q len	10.2	11.2	-	-	-	**11.4**
Ans. Q type	✓	✓	✗	✗	✓	✓
Amb. Q type	✗	✓	✓	✓	✓	✓
Una. Q type	✗	✓	✓	✗	✗	✓
Imp. Q type	✗	✓	✗	✗	✗	✓

The numbers are computed based on the total number of user utterances in the train set (if included), consistent with previous works. SParC [12] and CoSQL [17] are multi-turn datasets, while NoisySP [7], AmbiQT [26] and AMBROSIA [9] are single-turn datasets.

Additionally, the Response Quality Score (RQS) measures the quality of the model's natural language responses.

TDEX is a comprehensive metric that evaluates the accuracy of user query type classification and execution accuracy. For a set of N questions, where C_i denotes the expected classification and $\hat{C}_i$ represents the predicted classification for the i-th question, S_i denotes the ground truth SQL query and $\hat{S}_i$ represents the predicted SQL query for the i-th question, TDEX is computed as:

$$\text{TDEX} = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} \varepsilon_{\text{exec}}(S_i, \widehat{S}_i) & \text{(a)} \\ \delta_{\text{type}}(C_i, \widehat{C}_i) & \text{(b)} \end{cases} \tag{1}$$

(a) $C_i = '\text{Answerable}'$ *or* $'\text{Ambiguous}'$
(b) otherwise

where $\varepsilon_{\text{exec}} = 1$ if the execution result of $\hat{S}_i$ matches the execution result of S_i, and $\varepsilon_{\text{exec}} = 0$ otherwise; $\delta_{\text{type}} = 1$ if $\hat{C}_i$ matches C_i, and $\delta_{\text{type}} = 0$ otherwise.

Table 2 Correlation analysis between human and GPT-4 ratings for responses to different types of questions where SQL answer is not applicable.

Type	Pearson	P-value	Spearman	P-value	Kendall	P-value
Una.	0.86	5.44e-07	0.57	7.63e-03	0.54	8.76e-03
Amb.	0.96	4.07e-08	0.88	3.09e-05	0.81	3.86e-04
Imp.	0.71	3.46e-23	0.62	1.07e-16	0.61	7.14e-14

To evaluate the quality of a model's responses, we use an LLM-assisted rating method assessing Utility, Accuracy, Completeness, Clarity, and Relevance on a 0-to-2 scale, with a maximum score of 10. This approach, where LLMs directly score responses, has been demonstrated to be superior in assessing answer quality compared to traditional metrics like BLEU, ROUGE, and BERT-Score. [32, 33]. We employ GPT-4o-mini due to its strong alignment with human judgment [34, 35, 36] and enhance evaluation calibration using the Multiple Evidence Calibration methodology [37]. To ensure alignment with human judgment, three experts scored each of 100 samples, and their scores were averaged. Pearson, Spearman, and Kendall's Tau correlations were calculated, confirming a strong positive correlation between LLM and human scores, as shown in Table 2, demonstrating its reliability.

3.5 Baselines

We evaluate eight popular LLMs' performance, including close-source and open-source options. The close-source LLMs evaluated are GPT-4 Turbo[1], GPT-3.5 Turbo[2], and Gemini-1.5 Flash[3]. The open-source LLMs include Llama-3 (70B and 8B)[4], Llama-3-SQLCoder-8B[5] (a Llama-3 model specifically trained in the text-to-SQL domain), Codellama-7B[6], and Mistral-7B-v0.2[7]. See Appendix A for experimental details.

3.6 Result of MMSQL Evaluation

Table 3 presents the performance of the evaluated LLMs in the zero-shot evaluation on multi-turn text-to-SQL tasks of the MMSQL test set. This test set encompasses four types of questions. We assessed the models based on overall performance (TDEX), SQL generation performance (EX and EM), and their ability to recognize different types of questions. We provide a detailed discussion of the results in the following sections.

Comparative Performance of Closed-Source and Open-Source LLMs

As shown in Table 3, GPT-4 Turbo demonstrated exceptional performance with a TDEX score of 67.0, while Gemini-1.5 Flash scored 65.8. GPT-4 Turbo also achieved the highest RQS of 5.8, indicating its strong overall capabilities. Meanwhile, Llama3-70B showed comparable TDEX performance to GPT-3.5, scoring 62.8 and 64.1 respectively, highlighting the robust potential of closed-source models in zero-shot settings. Notably, Llama3-8B achieved a TDEX of 64.0, slightly surpassing GPT-3.5.

Regarding question type recognition, GPT-4 achieved the highest average F1 score of 68.2. Open-source models also performed commendably, with Llama3-8B attaining an average F1 score of 64.2, demonstrating their competitive edge. While closed-source models exhibit superior performance across many metrics, open-source models are rapidly catching up.

[1] https://platform.openai.com/docs/models/GPT-4Turbo-and-gpt-4
[2] https://platform.openai.com/docs/models/gpt-3-5-turbo
[3] https://deepmind.google/technologies/gemini
[4] https://llama.meta.com/llama3
[5] https://defog.ai
[6] https://www.llama.com/docs/integration-guides/meta-code-llama
[7] https://mistral.ai

Table 3 Performance of the models on MMSQL test set.

Model	TDEX	IEX	EX	EM	RQS	Ans.		Una.		Amb.		Imp.		F1
						Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec	
GPT-4 Turbo	**67.0**	**30.2**	**70.0**	51.0	**5.80**	90.3	89.8	25.9	70.0	56.9	38.4	**100.0**	98.0	**68.2**
GPT-3.5 Turbo	64.1	25.5	69.6	47.3	4.74	88.8	89.2	16.0	65.0	64.3	20.9	**100.0**	96.0	61.1
Gemini-1.5 Flash	65.8	30.1	**70.0**	**52.3**	4.03	85.9	95.8	26.3	50.0	58.3	8.1	**100.0**	95.4	59.3
Llama3-70B	62.8	22.8	66.4	47.4	3.86	84.9	95.2	27.5	55.0	80.0	9.3	**100.0**	92.7	59.8
Llama3-8B	64.0	20.1	66.1	45.7	4.55	88.3	93.2	21.4	60.0	83.3	23.3	**100.0**	96.7	64.2
SQLCoder-8B	30.7	24.8	63.2	31.0	3.43	84.6	**99.6**	**77.8**	35.0	0.0	0.0	**100.0**	98.0	59.7
Codellama-7B	30.7	3.4	27.2	21.7	5.09	**93.9**	16.5	4.3	**85.0**	96.6	**66.3**	56.8	**100.0**	46.9
Mistral-7B-v0.2	26.4	0.7	13.7	11.2	4.37	82.1	57.6	4.7	55.0	**100.0**	50.0	79.1	77.5	55.3

The best score in each column is highlighted in **bold**, and the second-highest score is underlined. Additionally, the models' abilities to recognize Answerable, Unanswerable, Ambiguous, and Improper questions are evaluated, with precision and recall provided for each category. **F1** represents the average F1 score across all categories. All in % except for average RQS.

Inferior Performance in Specific Question Types

As shown in Table 3, both open-source and closed-source models face significant challenges with unanswerable and ambiguous questions, with the latter being particularly difficult. For example, despite demonstrating strong overall performance, GPT-4 Turbo, which achieves the highest question type F1 score of 68.2, exhibits much lower precision and recall for unanswerable questions (56.9 and 38.4) and ambiguous questions (25.9 and 70.0) compared to other types. This pattern is also consistent across other models, highlighting a widespread difficulty in accurately distinguishing such question types. In terms of natural language response quality, as shown in Table 4, Gemini-1.5 Flash performs well on unanswerable questions with a score of 9.78, but significantly drops to 4.31 on ambiguous questions. Codellama-7B is the lowest-performing model overall, with an average score of 5.54, particularly struggling with ambiguous questions at a score of only 3.57. This trend underscores a common challenge all models face in interpreting and responding to ambiguous queries, indicating a need for further development in handling ambiguity.

Table 4 Response quality scores of models in handling different types of questions.

Model	Una.	Amb.	Imp.	Ave.
GPT-4 Turbo	<u>9.38</u>	**7.00**	**9.43**	**8.60**
GPT-3.5 Turbo	8.38	<u>4.73</u>	8.29	7.13
Gemini-1.5 Flash	**9.78**	4.31	<u>8.94</u>	<u>7.68</u>
Llama3-70B	8.80	4.60	8.38	7.26
Codellama-7B	4.75	3.57	8.31	5.54

The best score in each column is highlighted in **bold**, and the second-highest score is <u>underlined</u>.

Clarification for Ambiguous Queries

As illustrated in Figure 2, several representative models exhibit a notable decline in execution accuracy when addressing ambiguous queries, compared to their performance with clear, answerable queries. For example, GPT-4 Turbo's execution accuracy decreases from 51.0% with answerable queries to 41.3% when handling ambiguous ones. Similarly, GPT-3.5 Turbo and Llama3-70B show significant declines, with accuracy falling from 47.3% to 34.5% and from 47.4% to 34.7%, respectively. However, execution accuracy improves significantly when models detect ambiguity and prompt users for clarification. For instance, when ambiguous queries are clarified, GPT-3.5 Turbo's query match accuracy rises from 34.5% to 49.0%. This suggests that generating SQL for ambiguous queries without clarification results in diminished performance. Therefore, it is crucial for models to not only generate SQL queries but also effectively communicate any ambiguities to users. By clarifying uncertainties and helping users understand them, models can assist in obtaining accurate information. By incorporating the clarification process across multiple interactions, models can formulate more precise SQL queries, enhancing text-to-SQL systems' usability and effectiveness.

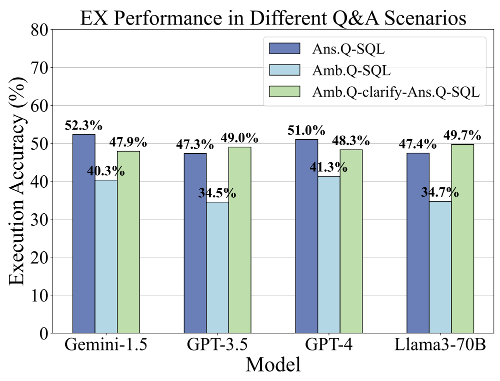


Fig. 2 EX performance analysis of different Q&A scenarios in model outputs of MMSQL test set.

4 Methodology

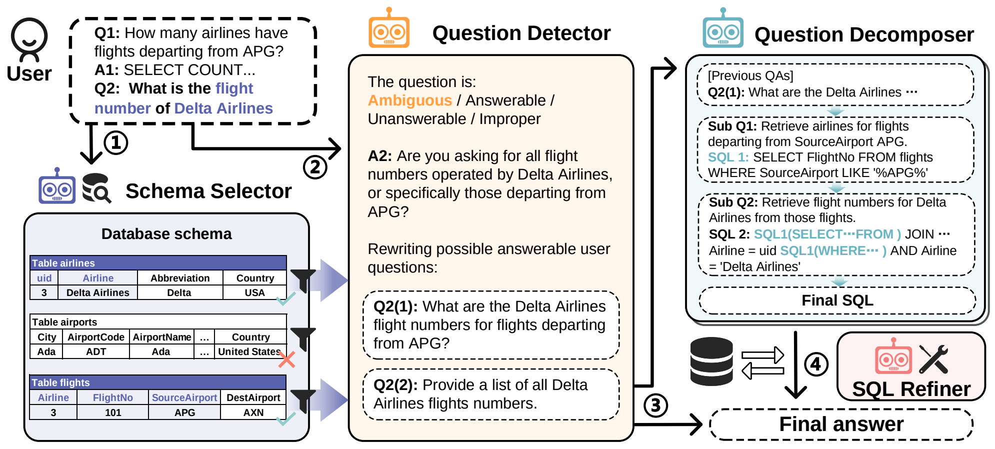

User
Q1: How many airlines have flights departing from APG?
A1: SELECT COUNT...
Q2: What is the flight number of Delta Airlines
①
②
Schema Selector
Database schema
Table airlines
uid | Airline | Abbreviation | Country
3 | Delta Airlines | Delta | USA
Table airports
City | AirportCode | AirportName | ... | Country
Ada | ADT | Ada | ... | United States
Table flights
Airline | FlightNo | SourceAirport | DestAirport
3 | 101 | APG | AXN
Question Detector
The question is:
Ambiguous / Answerable / Unanswerable / Improper
A2: Are you asking for all flight numbers operated by Delta Airlines, or specifically those departing from APG?
Rewriting possible answerable user questions:
Q2(1): What are the Delta Airlines flight numbers for flights departing from APG?
Q2(2): Provide a list of all Delta Airlines flights numbers.
③
Question Decomposer
[Previous QAs]
Q2(1): What are the Delta Airlines ···
Sub Q1: Retrieve airlines for flights departing from SourceAirport APG.
SQL 1: SELECT FlightNo FROM flights WHERE SourceAirport LIKE '%APG%'
Sub Q2: Retrieve flight numbers for Delta Airlines from those flights.
SQL 2: SQL1(SELECT···FROM) JOIN ··· Airline = uid SQL1(WHERE···) AND Airline = 'Delta Airlines'
Final SQL
④
SQL Refiner
Final answer

Fig. 3 The overview of our multi-agent framework comprises four components: (i) the Schema Selector, which narrows down the database schema to focus on relevant tables, reducing noise from irrelevant data; (ii) the Question Detector, which determines its type and reformulates it if the question is deemed potentially ambiguous, potentially generating multiple possible rewrites; (iii) the Question Decomposer, which breaks down complex questions into simpler, manageable sub-questions and applies chain-of-thought reasoning to solve them progressively; and (iv) the SQL Refiner, which utilizes an external tool for SQL execution, gathers feedback, then refines faulty SQL queries.

As illustrated in Figure 3, we introduce an LLM-based multi-agent collaborative framework that utilizes intelligent agents with distinct functionalities to enhance text-to-SQL parsing. This framework comprises a Question Detector agent, which identifies the type of question and determines the appropriate response strategy. Accompanying it is the Question Decomposer agent, responsible for generating SQL queries by breaking down complex questions into simpler sub-questions. Additionally, two auxiliary agents: the Schema Selector and the SQL Refiner aid in refining the process. The Schema Selector narrows down the database schema to focus on relevant tables, while the SQL Refiner uses external tools for SQL execution and feedback, ensuring the queries are accurate and effective. A detailed introduction of these agents will be presented.

4.1 Schema Selector

The Schema Selector is designed to identify the essential subset of a database schema necessary for answering a given question. By selectively focusing on the most relevant tables and columns, the Selector curtails the interference from extraneous data, enhancing operational accuracy. This retrieving capability is particularly pertinent in complex text-to-SQL systems, such as Business Intelligence (BI) platforms, which often interact with large databases containing numerous tables and columns [2, 38, 39]. In our framework, the Selector is activated to choose tables and columns relevant to the interaction record solely when the schema size exceeds a specific threshold; otherwise, the full schema is employed. This adaptive mechanism ensures that our framework maintains efficiency and effectiveness across different database scales and complexities.

4.2 Question Detector

The Question Detector is responsible for detecting the question type and deciding on an appropriate answering strategy, whether to directly answer or attempt SQL generation, based on the subset of the database schema and the question-answer history.

For answerable questions, the system proceeds to the Question Decomposer for further processing. In the case of ambiguous questions, where the query cannot be precisely mapped to the database schema, the system explains the ambiguity and attempts to rewrite it into answerable forms. For example, as shown in Figure 3, the question ”What is the flight number of Delta Airlines?” (Q2) could be interpreted in multiple ways: it might refer to flights departing from APG, as inferred from the previous context (Q1), or it could pertain to all flights in the database. Instead of rejecting such ambiguous queries, the system attempts to rewrite them into answerable questions and provides multiple potential answers in the final response.

For unanswerable questions, where the query is not ambiguous but cannot be answered due to data limitations or inappropriateness, the system informs the user of the reason. For improper questions, the LLM is tasked with providing a helpful response to assist the user as much as possible. In both scenarios, the system aims to guide the user towards obtaining a meaningful and helpful response as the final answer.

4.3 Question Decomposer

The Question Decomposer plays a critical role in the multi-agent framework by breaking down complex questions into simpler, manageable Sub-Questions $\{Q_1, Q_2, \ldots, Q_{|Q|}\}$. For each Sub-Question Q_i, it generates a corresponding sub-SQL S_j based on $\mathrm{S}_{<j}$, and the schema subset to address specific parts of the original query.

This process involves applying chain-of-thought reasoning, allowing the system to progressively solve each sub-question. As shown in Fig. 3, the Question Decomposer takes an ambiguous question, "What are the Delta Airlines flight numbers for flights departing from APG?", and decomposes it into sub-questions like "Retrieve airlines for flights from SourceAirport APG" and "Retrieve flight numbers for Delta Airlines from those flights." Each sub-question is then translated into a sub-SQL, collectively forming the final SQL output.

4.4 SQL Refiner

The SQL Refiner, as an integral auxiliary operation of the Question Decomposer, serves a critical role in ensuring the precision and reliability of the SQL queries generated by our multi-agent framework. This component is indispensable for the inspection and correction of the responses produced, particularly when addressing the complex demands of text-to-SQL tasks. Frequently utilized, the Refiner has demonstrated its effectiveness in enhancing the output to meet stringent accuracy requirements [5, 6, 31]. Drawing a parallel to its counterparts in software development, such as metaGPT [30] and ChatDev [40], where intelligent agents are tasked with overall architectural design, code writing, and comprehensive testing, the SQL Refiner plays a similar role in the domain of database interactions. It adeptly adjusts SQL queries to accommodate various datasets, database schemas, and SQL generation styles, thereby ensuring the accuracy and effectiveness of the generated queries.

5 Experiments

5.1 Experimental Setup

We conducted a comprehensive experiment utilizing the MMSQL test suite to rigorously evaluate the efficacy of our multi-agent framework in managing intricate multi-turn text-to-SQL tasks. The experimental design was meticulously crafted to evaluate the framework's proficiency in producing valid SQL and coherent natural language responses, alongside its capacity to categorize diverse question types effectively. To quantitatively assess the quality of the natural language responses, we employed a suite of metrics, including the average RQS, EM, average F1 score across all categories, and TDEX, to provide a holistic evaluation of the models' performance.

We conduct experiments on three prominent models: GPT-4 Turbo, Gemini-1.5 Flash, and Llama3-70b, as delineated in Section 3.6. These models were selected as our baselines to underscore the comparative effectiveness of our methodology. The objective of our experiments was to conduct a meticulous analysis elucidating how the incorporation of the multi-agent framework augments the performance of these models across a spectrum of evaluation metrics.

5.2 Overall Performance

Table 5 illustrates the performance of our method and baseline models on the MMSQL test set, highlighting the enhancements achieved by integrating our multi-agent framework with different models. The results demonstrate the benefits of our approach, with

Table 5 Performance metrics comparison on the MMSQL test set, demonstrating the improvements achieved by integrating the multi-agent framework with different models. Specific improvements are indicated in parentheses.

Model	TDEX	EX	RQS	F1 Score
GPT-4 Turbo	67.0	70.0	5.80	68.2
Multi-Agent+GPT-4 Turbo	70.0(+3.0)	74.4(+4.4)	5.98(+0.18)	69.5(+1.3)
Gemini-1.5 Flash	65.8	70.0	4.03	59.3
Multi-Agent+Gemini-1.5 Flash	69.4(+3.6)	73.7(+3.7)	**7.05(+2.02)**	70.7(+11.4)
Llama3-70B	62.8	66.4	3.86	59.8
Multi-Agent+Llama3-70B	**70.7(+7.9)**	**74.8(+8.4)**	6.64(+2.78)	**70.8(+11.0)**

The best values are marked in **bold**.

significant improvements across all metrics for each model. For example, the TDEX score increases from 62.8 to 70.7 and the F1 Score improves from 59.8 to 70.8 for Llama3-70B, showcasing the robustness of our framework in generating accurate SQL queries and natural language responses. These improvements underscore the effectiveness of our multi-agent framework in bolstering the capabilities of existing models to handle complex text-to-SQL tasks on the MMSQL benchmark.

5.3 Ablation Study

In Table 6, we presents the findings from an ablation study on the MMSQL test set, examining the impact of removing individual components from the multi-agent framework when integrated with Gemini-1.5 Flash. The complete model achieves its best performance with scores of 69.4 in TDEX, 73.7 in EX, 7.05 in Average RQS, and an F1 Score of 70.7. The study reveals that the removal of any component results in a decrease in performance across these metrics.

Table 6 Ablation study examining the impact of each component within the multi-agent framework on the model's performance on the MMSQL test set.

Model	TDEX	EX	Average RQS	F1 Score
Multi-Agent + Gemini-1.5 Flash	**69.4**	**73.7**	**7.05**	**70.7**
w/o selector	68.6	73.4	6.43	66.2
w/o detector	66.1	70.8	5.57	64.4
w/o decomposer	68.6	73.1	6.33	68.1
w/o refiner	67.6	70.5	**7.05**	**70.7**

The best values are marked in **bold**.

Notably, the absence of the Detector results in the poorest performance with a sharp decline to 5.57 in Average RQS and 64.4 in F1 Score, indicating its pivotal role in selecting answering strategies. The Selector and Decomposer also contribute notably to the model's accuracy and response quality. The Refiner, also shows a notable impact when removed, particularly affecting the EX, dropping to 70.5, suggesting its role in correcting errors is indispensable for ensuring the accuracy of the final SQL queries generated.

6 Discussion

In this study, we introduced the MMSQL test suite, a pioneering benchmark specifically designed to evaluate LLMs across complex multi-turn and multi-type text-to-SQL tasks. Our comprehensive evaluation of eight popular LLMs revealed that even the most advanced models struggle significantly with MMSQL's diverse question types, particularly with ambiguous or unanswerable queries. This difficulty underscores the urgent need for models that possess enhanced capabilities for intent understanding and uncertainty management, as directly responding to ambiguous queries with SQL can severely degrade overall performance. Our findings are detailed in Table 3, highlighting the impressive potential of open-source models. Notably, the Llama3 series demonstrates performance that matches or even exceeds that of closed-source models on several metrics. These results suggest pathways for developing more reliable and robust text-to-SQL systems. Furthermore, we designed a multi-agent framework tailored for handling multi-type text-to-SQL problems, which distinguishes itself from previous approaches that opted for abstention in the face of ambiguous queries [10, 7]. By refining the existing text-to-SQL multi-agent framework [5, 6] and incorporating a Question Detector Agent, our framework attempts to resolve ambiguities by requesting clarifications from users. This innovative approach not only enhances the model's ability to deal with complex queries but also significantly improves user interaction. These advancements represent a significant step forward in the evolution of text-to-SQL technologies, demonstrating notable progress across a range of evaluation metrics.

Our study has highlighted advancements in using LLMs for text-to-SQL applications, yet it also reveals several limitations that point to opportunities for further development. Firstly, the performance of the open-source LLMs used in our tests could be enhanced by using larger open-sourced LLMs. Secondly, our multi-agent framework is currently based on closed-source LLMs. Exploring open-source, domain-specific LLMs and further decomposing and annotating the MMSQL training set could enhance our framework's effectiveness synergistically. This approach would provide tailored training data for each agent, allowing for targeted fine-tuning that leads to better results. This dataset also does not incorporate domain-specific knowledge bases, which limits its effectiveness in specialized fields such as medicine, law, or finance. Addressing these issues by integrating domain-specific knowledge into the training process could improve the model's ability to accurately process and respond to specialized queries.

7 Conclusion

Existing text-to-SQL evaluation methods often focus on increasing the complexity of SQL queries to enhance model performance. However, these approaches frequently encounter significant challenges in handling conversational dynamics, which further impede their development. In response to these challenges, this paper introduces the MMSQL test suite and proposes a novel multi-agent framework. The MMSQL test suite provides a structured approach to assess the capabilities of LLMs and offers an environment with diverse question types from a multi-turn perspective. Our analysis and experiments based on MMSQL illuminate the current limitations of LLMs. The multi-agent framework adeptly manages diverse question types. Ultimately, our work contributes to the democratization of data sharing, enhances the utility and robustness of text-to-SQL systems, and provides practical guidance for achieving high performance in this domain.

Funding. National Natural Science Foundation, China (No. 62172123), the Key Research and Development Program of Heilongjiang (Grant No. 2022ZX01A36).

Data availability. The data availability is outlined in Appendix B.

Code availability. The code used in this article is publicly available on GitHub together with instructions on how to use the code.

Declarations

Supplementary information. The MMSQL dataset, experimental results, and the code for generating the training set are available in the Supplementary Files.

Conflict of interest. The Authors listed in this article declare that they have no conflict of interest.

Ethical approval. We prioritize ethical considerations and data privacy in our work. All content and labels used for our data augmentation and manual annotation are sourced exclusively from publicly available datasets.

Appendix A Implementation Details

In all experiments, we employed the same chatting format for each LLM across all experiments. We applied greedy decoding strategies across all LLMs during inference and evaluation to ensure reproducibility. The experiments were performed on a server with an Intel(R) Xeon(R) Gold 6133 CPU @ 2.50GHz and 4 NVIDIA A800 80GB PCIe GPU.

Appendix B Availability of data and material

We acknowledge the use of the CoSQL[8] and SParC[9] datasets, both licensed under the CC BY-SA 4.0 license[10]. This license allows for the sharing and adaptation of the datasets, provided that appropriate credit is given to the original creators and any modifications are distributed under the same license.

[8]CoSQL datasets from [17] https://yale-lily.github.io/cosql
[9]SParC datasets from [12] https://yale-lily.github.io/sparc
[10]https://creativecommons.org/licenses/by-sa/4.0/

The MMSQL repository[11] provides access to the augmented datasets generated by QDA-SQL, testing scripts, and experimental records. The implementation of the QDA-SQL method is available in a separate repository[12].

[11]MMSQL dataset and associated scripts are accessible at https://github.com/mcxiaoxiao/MMSQL
[12]QDA-SQL scripts are accessible at https://github.com/mcxiaoxiao/QDA-SQL

References

[1] Guanming Xiong, Junwei Bao, Hongfei Jiang, Yang Song, and Wen Zhao. Interactive-t2s: Multi-turn interactions for text-to-sql with large language models, 2024.

[2] Jinqing Lian, Xinyi Liu, Yingxia Shao, Yang Dong, Ming Wang, Zhang Wei, Tianqi Wan, Ming Dong, and Hailin Yan. Chatbi: Towards natural language to complex business intelligence sql, 2024.

[3] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36, 2024.

[4] Dingzirui Wang, Longxu Dou, Wanxiang Che, Jiaqi Wang, Jinbo Liu, Lixin Li, Jingan Shang, Lei Tao, Jie Zhang, Cong Fu, et al. Conda: state-based data augmentation for context-dependent text-to-sql. *International Journal of Machine Learning and Cybernetics*, pages 1–12, 2024.

[5] Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O. Arik. Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql, 2024.

[6] Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Linzheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. Mac-sql: A multi-agent collaborative framework for text-to-sql, 2024.

[7] Bing Wang, Yan Gao, Zhoujun Li, and Jian-Guang Lou. Know what i don't know: Handling ambiguous and unknown questions for text-to-sql. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5701–5714, 2023.

[8] Vipula Rawte, Amit Sheth, and Amitava Das. A survey of hallucination in large foundation models, 2023.

[9] Irina Saparina and Mirella Lapata. Ambrosia: A benchmark for parsing ambiguous questions into database queries, 2024.

[10] Gyubok Lee, Woosog Chay, Seonhee Cho, and Edward Choi. Trustsql: A reliability benchmark for text-to-sql models with diverse unanswerable questions. *arXiv preprint arXiv:2403.15879*, 2024.

[11] Xinyi He, Mengyu Zhou, Xinrun Xu, Xiaojun Ma, Rui Ding, Lun Du, Yan Gao, Ran Jia, Xu Chen, Shi Han, Zejian Yuan, and Dongmei Zhang. Text2analysis: A benchmark of table question answering with advanced data analysis and unclear queries, 2023.

[12] Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, et al. Sparc: Cross-domain semantic parsing in context. *arXiv preprint arXiv:1906.02285*, 2019.

[13] Jiaqi Guo, Ziliang Si, Yu Wang, Qian Liu, Ming Fan, Jian-Guang Lou, Zijiang Yang, and Ting Liu. Chase: A large-scale and pragmatic chinese dataset for cross-database context-dependent text-to-sql. 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 2316–2331, 2021.

[14] 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.

[15] Yinggang Sun, Ziming Guo, Haining Yu, Chuanyi Liu, Xiang Li, Bingxuan Wang, Xiangzhan Yu, and Tiancheng Zhao. Qda-sql: Questions enhanced dialogue augmentation for multi-turn text-to-sql. *arXiv preprint arXiv:2406.10593*, 2024.

[16] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning, 2017.

[17] Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China, November 2019. Association for Computational Linguistics.

[18] Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. A comprehensive exploration on wikisql with table-aware word contextualization, 2019.

[19] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *arXiv preprint arXiv:2109.05093*, 2021.

[20] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online, July 2020. Association for Computational Linguistics.

[21] Luca Sala, Giovanni Sullutrone, and Sonia Bergamaschi. Text-to-sql with large language models: Exploring the promise and pitfalls. In Maurizio Atzori, Paolo Ciaccia, Michelangelo Ceci, Federica Mandreoli, Donato Malerba, Manuela Sanguinetti, Antonio Pellicani, and Federico Motta, editors, *Proceedings of the 32nd Symposium of Advanced Database Systems, Villasimius, Italy, June 23rd to 26th, 2024*, volume 3741 of *CEUR Workshop Proceedings*, pages 260–270. CEUR-WS.org, 2024.

[22] Karime Maamari, Fadhil Abubaker, Daniel Jaroslawicz, and Amine Mhedhbi. The death of schema linking? text-to-sql in the age of well-reasoned language models, 2024.

[23] Mahdi Mostajabdaveh, Timothy T Yu, Rindranirina Ramamonjison, Giuseppe Carenini, Zirui Zhou, and Yong Zhang. Optimization modeling and verification from problem specifications using a multi-agent multi-stage llm framework. *INFOR: Information Systems and Operational Research*, pages 1–19, 2024.

[24] Mohammadreza Pourreza and Davood Rafiei. Din-sql: Decomposed in-context learning of text-to-sql with self-correction, 2023.

[25] 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.

[26] Adithya Bhaskar, Tushar Tomar, Ashutosh Sathe, and Sunita Sarawagi. Benchmarking and improving text-to-SQL generation under ambiguity. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7053–7074, Singapore, December 2023. Association for Computational Linguistics.

[27] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[28] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents, 2023.

[29] Significant Gravitas. AutoGPT. https://github.com/Significant-Gravitas/AutoGPT, 2023. MIT License.

[30] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2024.

[31] Wenxuan Xie, Gaochen Wu, and Bowen Zhou. Mag-sql: Multi-agent generative approach with soft schema linking and iterative sub-sql refinement for text-to-sql, 2024.

[32] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023.

[33] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore, December 2023. Association for Computational Linguistics.

[34] Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. Mt-eval: A multi-turn capabilities evaluation benchmark for large language models. *arXiv preprint arXiv:2401.16745*, 2024.

[35] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.

[36] Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. Mt-eval: A multi-turn

capabilities evaluation benchmark for large language models, 2024.

[37] Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators, 2023.

[38] Zhongyuan Wang, Richong Zhang, Zhijie Nie, and Jaein Kim. Tool-assisted agent on sql inspection and refinement in real-world scenarios, 2024.

[39] Nina Narodytska and Shay Vargaftik. Lucy: Think and reason to solve text-to-sql, 2024.

[40] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Chatdev: Communicative agents for software development, 2024.