GBV-SQL: 引导生成和 SQL2Text 回译验证 用于多代理文本到 SQL 转换

Daojun Chen[1*] Xi Wang[2] Shenyuan Ren[3] Qingzhi Ma[1†] Pengpeng Zhao[1] An Liu[1]

[1]School of Computer Science & Technology, Soochow University, Suzhou, China
[2]Computer Science, University of Sheffield, UK
[3]School of Computer Science and Technology, Beijing Jiaotong University, China
djchen@stu.suda.edu.cn, xi.wang@sheffield.ac.uk, syren@bjtu.edu.cn, qzma@suda.edu.cn, ppzhao@suda.edu.cn, anliu@suda.edu.cn

Abstract

尽管大型语言模型显著提升了 Text2SQL 生成的能力，但在语义理解上仍存在一个关键差距，即语法正确的查询往往误解了用户的真实意图。为缓解这一挑战，我们提出了 GBV-SQL，这是一种新颖的多代理框架，引入了带有 SQL2Text 回译验证的引导式生成。该机制利用专门的代理将生成的 SQL 翻译回自然语言，以验证其逻辑是否与原始问题一致。至关重要的是，我们的研究表明当前评估受到系统性问题的影响：基准测试本身的质量低下。我们引入了“黄金错误”的正式分类，这些是真实数据中存在的广泛缺陷，并展示了它们如何掩盖模型的真实性能。在具有挑战性的 BIRD 基准上，GBV-SQL 实现了 63.23%的执行准确率，绝对提升了 5.8%。移除有缺陷的例子后，在 Spider 基准上，GBV-SQL 达到了 96.5%（开发）和 97.6%（测试）的执行准确率。我们的工作不仅提供了一个强大的语义验证框架，还对基准测试的完整性提出了批判性观点，强调了更严格的数据库整理需求。

1 介绍

Text2SQL 是一项具有挑战性的任务，涉及将自然语言问题（NLQ）自动转换为可以在关系数据库上执行的结构化查询语言（SQL）。(Zelle and Mooney 1996; Qin et al. 2022) 主要目标是使非技术用户能够仅使用自然语言与复杂数据库进行交互，消除学习 SQL 语法的障碍。作为自然语言处理和数据库研究领域长期的目标之一 (Nguyen et al. 2023; Zhu et al. 2023)，这一领域因大型语言模型（LLMs）的最新进展而发生了深刻变革。这些模型，特别是在上下文学习（ICL）范式下，现在代表了最先进的技术水平，通常表现出超越传统微调方法的性能。(Pourreza and Rafiei 2023; Li et al. 2023a; Liu et al. 2024)

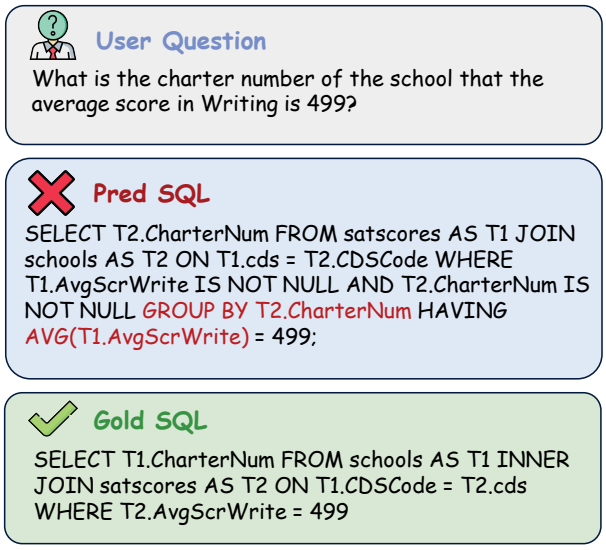

图 1: A BIRD 开发示例，其中语法上有效的 Pred SQL 对查询进行了语义上的误解：它不必要地按特许群组对“平均分数”进行聚合，而用户明确寻求一所分数为 499 的学校。

Despite this significant progress, a critical challenge persists: ensuring the generated SQL query is not only syntactically correct but also semantically faithful to the user's original intent. This challenge stems from the inherent semantic gap between an unstructured Natural Language Question (NLQ) and the for-

*First Author
†Corresponding Author

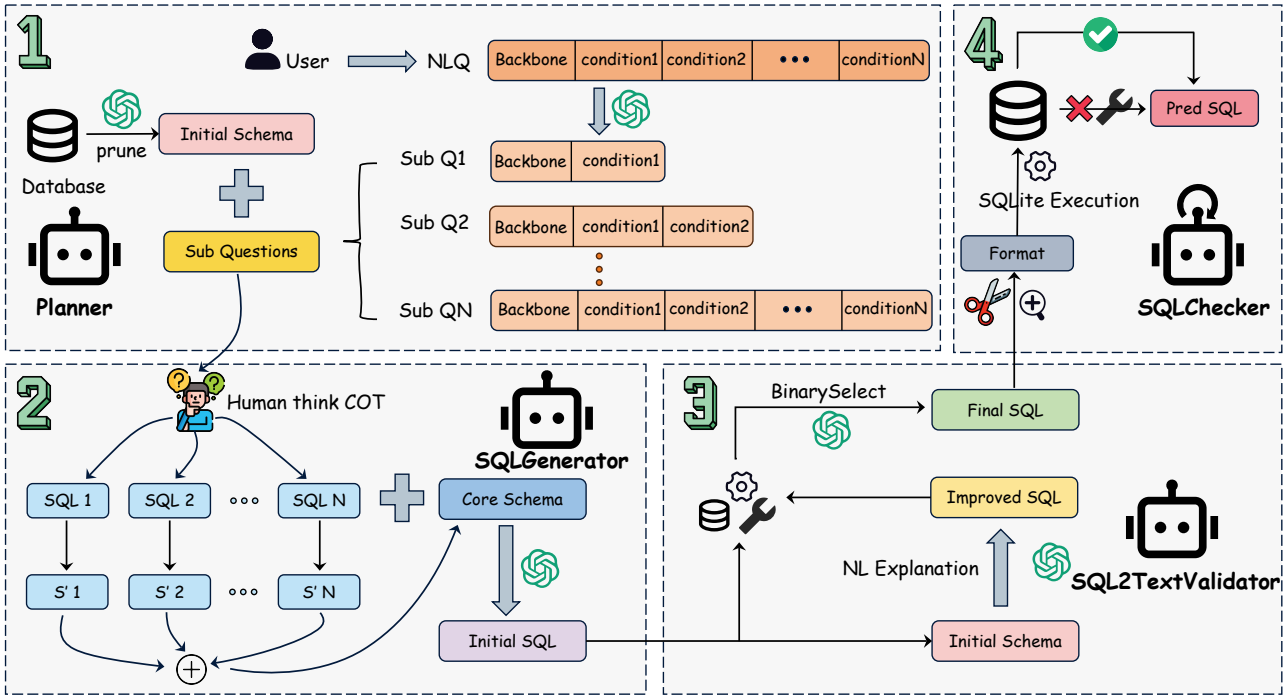

图 2: GBV-SQL 框架的工作流程，包括四个代理：(1) 规划者 (2)SQL 生成器 (3)SQL2 文本验证器 (4)SQL 检查器。

mal structure of SQL, which remains a primary source of failure. A syntactically valid query can still misinterpret the user's goal, producing plausible but erroneous results that are difficult to detect. For instance, as illustrated in Figure 1, when asked for a school whose "average score" column has a value of 499, a model might instead generate a query that performs an unnecessary aggregation, calculating the average of the "average score" column across a group of schools and checking if that new result is 499. This error, transforming a direct filtering condition ('WHERE') into a flawed aggregation condition ('HAVING'), highlights the inadequacy of syntax-only verification. While existing state-of-the-art methods employ techniques like query decomposition (Gao et al. 2024a; Xie, Wu, and Zhou 2024) and multi-agent collaboration (Wang et al. 2025) to ensure that LLMs generate reasonable SQL, they often lack a dedicated mechanism to robustly validate the final query's logic against the NLQ's semantics.

为了弥合这一语义差距，我们提出了 GBV-SQL，一个多智能体框架，集成了引导生成与 SQL2Text 反向翻译验证。工作流由四个专门的代理（图 2）编排。过程始于一个规划者，它修剪数据库模式并将 NLQ 分解为更简单的子问题。然后，一个 SQL 生成器使用新颖的人类样链条思维过程构建查询。关键的是，为了确保语义保真度，一个专门的 SQL2 文本验证器执行反向翻译，在发现与用户意图逻辑不符时纠正查询以生成改进版本。最后，一个 SQL 检查器进行最终检查，启动迭代修复循环以解决任何语法或执行错误，直到查询有效为止。整个过程旨在系统地针对并最小化不准确性。

我们在 Spider 基准测试 (Yu et al. 2018) 上的严格评估产生了第二个同样重要的发现：我们模型看似失败的很大一部分并不是源于我们的方法，而是由于基准测试的真实数据中存在的内在错误。我们将这些问题称为"黄金错误"，并提出了一种正式的分类学来对它们进行系统的分类。这一发现从根本上挑战了当前评估实践的可靠性，并突显了该领域的一个系统性问题。

我们的主要贡献有三点：

- 我们提出了 GBV-SQL，一个新颖的多代理框架，该框架引入了 SQL2Text 反向翻译验证机制，以明确弥合 Text2SQL 中的语义差距，确保生成的查询逻辑忠实于用户的原始意图。
- 我们对广泛使用的 Spider 基准测试中的错误进行了系统性分析，引入了一种新型的"黄金错误"分类法，这些错误是地面真实数据本身固有的缺陷。
- GBV-SQL 在具有挑战性的 BIRD 基准测试中实现了 63.23%的执行准确性，绝对提升了 5.8%。在一个

经过修正的“干净”版本的 Spider 上，纠正了已识别的“Gold Errors”，我们的模型在开发集上的执行准确率达到 96.5%，在测试集上达到 97.6%，这揭示了其真实能力和基准质量的关键影响。

2 相关工作

基于 ICL 的方法用于文本到 SQL 转换 大型语言模型（LLMs）中的上下文学习（ICL）是当前文本到 SQL（Text2SQL）的最先进范式。解决复杂查询的主要策略是分解，即将一个困难的问题分解为更简单的子问题。例如，DIN-SQL(Pourreza and Rafiei 2023) 首次应用 CoT 提示进行分解。DAIL-SQL(Gao et al. 2024a) 然后在此框架中探索了先进的提示设计。Tai 等人评估了多种提示策略，并引入 QDecomp 逐步向子问题添加模式细节 (Tai et al. 2023)。为了提高整体鲁棒性，其他研究主要集中在两个方面。首先是通过更准确的模式链接来提升模型输入的质量 (Cao et al. 2024; Wang and Liu 2025; Li et al. 2024b)。第二个是通过事后纠正生成的 SQL(Dong et al. 2023; Ren et al. 2024) 或从多个候选者中选择最优查询来精炼模型的输出 (Lee et al. 2025; Pourreza et al. 2024)。尽管这些多样化的策略推动了该领域的进步，但它们主要在单向工作流程中运行。它们缺乏明确的机制来验证最终语法正确的 SQL 是否真正与用户的原始意图语义对齐。我们的工作直接针对这个语义差距引入了一个新的验证步骤。

多智能体系统用于文本到 SQL 转换 为了编排多步骤推理过程，最近的研究转向了多智能体系统。MAC-SQL(Wang et al. 2025) 和 MAG-SQL(Xie, Wu, and Zhou 2024) 通过使用多个基于 LLM 的代理来协作管理分解、SQL 生成和优化，开创了这一方法。这些框架展示了分工可以有效地处理复杂查询。然而，现有的代理只合作生成 SQL；没有一个执行最终查询的明确语义验证。其他基于代理的系统，如 SQLFixAgent(Cen et al. 2025)，专注于纠正微调模型产生的错误，这属于不同的任务背景。我们的 GBV-SQL 框架引入了一种新的代理原型：SQL2TextValidator。这个代理作为一个怀疑论者验证者，其唯一目的是执行 SQL 到文本的反向翻译。通过从逆向视角挑战生成查询的语义保真度，它提供了一个在先前协作模型中缺失的关键反馈循环。

基准质量和黄金错误 文本到 SQL 评估的可靠性本质上与基准数据集的质量息息相关。先前工作中错误分析已注意到地面真实标签中的缺陷，称为“黄金误差”(Wang et al. 2025; Lee et al. 2025)。进一步的研究深入探讨了具体质量问题，如分析标签准确性 (Renggli, Ilyas, and Rekatsinas 2025)，分类问题模糊性 (Dong et al. 2024) 或自动检测不正确的 SQL-NLQ 映射 (Yang et al. 2025)。这些努力凸显了对数据质量问题日益增长的认识。我们的工作在此基础上提出了一种正式的、结构化的黄金误差类型学。与之前的事后分析不同，我们的分类系统更加全面，将来自 SQL、NLQ 和数据库本身的错误进行归类。这一系统的框架不仅允许更精确且公平地重新评估 GBV-SQL 的真实能力，还为未来文本到 SQL 基准的数据整理和维护提供了一个有价值的工具。

```
Database ID: car_1
# Table: continents
(ContId, cont id [PRIMARY KEY]. Value examples:column type is
INTEGER, [1, 2, 3, 4].),
(Continent, continent. Value examples:column type is TEXT,
['europe', 'australia', 'asia', 'america'].)
# Table: countries
(CountryId, country id [PRIMARY KEY]. Value examples:column type is
INTEGER, [1, 2, 3, 4].),
(CountryName, country name. Value examples:column type is TEXT,
['usa', 'uk', 'sweden', 'russia'].),
(Continent, continent [FOREIGN KEY → continents.ContId]. Value
examples:column type is INTEGER, [2, 1, 5, 4].)
...
Foreign keys
countries.`Continent` = continents.`ContId`
...
```

图 3: 模式表示的提示模板。

3 方法论

3.1 概述

在本节中，我们介绍了 GBV-SQL，一个利用大规模语言模型的新型 Text2SQL 多智能体框架。如图 2所示，我们的框架包含四个代理：规划者、SQL 生成器、SQL2 文本验证器和 SQL 检查器。规划器通过调用一个大语言模型来修剪数据库模式到其最相关的元素，并将自然语言查询分解为子问题，从而启动该过程。模仿人类思维过程，SQL 生成器为每个子问题生成 SQL，然后再将其综合成最终的查询。SQL2Text 验证器通过提示一个大语言模型将完整的 SQL 翻译回自然语言并与原始 NLQ 进行比较，以确保语义完整性，从而产生一个优化后的查询。最后，SQL 检查器进行格式和语法检查，并纠正任何错误，以保证最终的 SQL 可执行。

3.2 规划者

模式表示 先前的工作 (Bogin, Berant, and Gardner 2019; Guo et al. 2019; Lei et al. 2020; Gao et al. 2024b) 表明，一个组织良好的数据库模式表示可以提高 LLMs 生成 SQL 的准确性。在本文中，我们基于 MAC-SQL 的方法来组织我们的模式描述，但进行了增强。我们包含了数据库列的数据类型，并提供了更直观的外键描述。

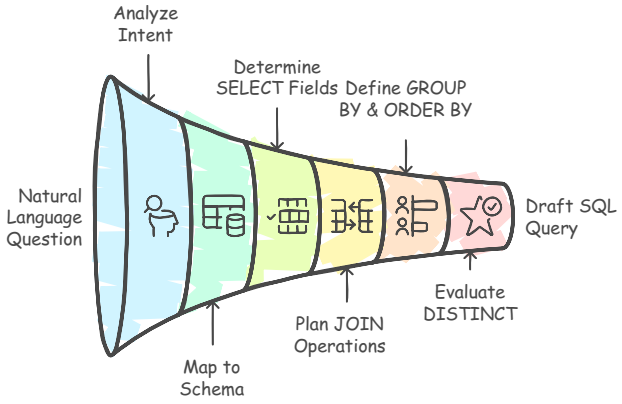

图 4: 类人思维链工作流用于起草 SQL 查询。

此外，我们是第一个在模式表示中包含主键描述的。具体提示设计如图 3所示。

模式剪枝 过长的模式可能导致 LLM 专注于与 NLQ 无关的信息并产生高额的 token 成本。因此，我们通过提示 LLM 对完整的数据库模式进行初步修剪。我们提示 LLM 分析 NLQ (Q) 和完整的数据库模式 (D)，仅保留最相关的信息。

$$S_{\text{initial}} = \text{Prune}(Q, D)\,, \tag{1}$$

其中 S_{initial} 代表与问题相关的修剪后的模式。

NLQ **分解** 如前所述的调查显示 (Pourreza and Rafiei 2023, 2024)，将问题分解为子问题是有效的策略。我们采用基于分治范式的分解策略，其中问题分解和 SQL 生成是分开的过程。具体来说，我们采用了 MAG-SQL(Xie, Wu, and Zhou 2024) 中提出的 Targets-Conditions 方法来根据其不同的条件从句和查询目标分解 NLQ，如图所示 2。

3.3 SQL 生成器

类人 Cot 为了模拟人类构思 SQL 查询的过程 (Yuan et al. 2025)，我们设计了一种新颖的 CoT 方法，如图 4 所示。该方法首先分析 NLQ 的意图。然后根据数据库模式识别相关的表和列，并确定要返回的具体信息。随后，它系统地分析如何构建必要的 SQL 子句，考虑连接的特定逻辑、分组和排序的列以及使用 DISTINCT 去重的可能性。最后，它强调在整个生成过程中遵守正确的 SQL 语法。

子 SQL 生成与合并 我们的框架采用两阶段过程生成 SQL。首先，对于由规划器分解的每个子问题，我们生成相应的子 SQL。随后，我们将这些子 SQL 中的上下文信息合并以合成最终答案。该过程首先应用我们的类人 CoT 方法，利用初始修剪后的模式 S_{initial} 为每个子问题（q_i）生成一个子 SQL。接下来的关键步骤是对每个生成的子 SQL（SQL_i）执行“回链”操作。此操作是一个 LLM 推理步骤，在该步骤中，模型被提示提取对子查询创建至关重要的精确表格-列对集（S_i）。

$$SQL_i = \text{HumanCoT}\big(S_{\text{initial}},\, q_i\big) \tag{2}$$

$$S_i = \text{BackLink}\big(SQL_i\big) \tag{3}$$

这些提取的模式子集随后被合并，以创建一个更加精细且高度相关的模式，S_{core}。这个新的模式比 S_{initial} 更准确，因为它直接源自回答每个子问题所需的 SQL 逻辑。合并操作定义为来自各个模式子集的所有表-列对的并集：

$$S_{\text{core}} = \bigoplus_{i=1}^{n} S_i \tag{4}$$

其中，对于模式中的任何表 t_k，运算符定义为：

$$\left(\bigoplus_{i=1}^{n} S_i\right)(t_k) = \bigcup_{i=1}^{n} S_i(t_k) \tag{5}$$

最后，这个综合模式 S_{core}，连同完整的子问题和子 SQL 对集合，为 LLM 生成最终的综合性查询提供了丰富且集中的上下文，在图 2中被称为初始 SQL。

3.4 SQL2 文本验证器

作为 GBV-SQL 的核心模块之一，SQL2TextValidator 通过提示 LLM 将 SQLGenerator 生成的初始 SQL 查询转换为详细的自然语言解释，执行语义验证。该解释描述了查询的整体功能及其组件的具体机制，特别关注查询的执行过程。然后，LLM 的任务是将此解释与原始 NLQ 进行比较，以识别任何语义差异。如果逻辑一致，则接受该查询；否则，LLM 会被提示更正 SQL，使其与用户的意图一致，从而产生经过验证的改进 SQL。为了最大限度地提高验证的有效性，我们结合了受 RSL-SQL(Cao et al. 2024) 启发的二元选择器。该选择器使用 LLM 评估初始 SQL 版本和验证后的 SQL 版本与原始 NLQ 之间的语义一致性，并考虑它们的执行结果以选择最终 SQL。

3.5 SQL 检查器

由于大型模型的幻觉性质，生成的 SQL 经常包含格式错误 (Shen et al. 2025)。因此，该模块引入了一个 SQL 格式修剪器。该组件根据两个关键原则优化查询：最小化原则，消除多余字段和冗余操作；以及最小可用性原则，避免侵入性格式（例如，四舍五入小数），除非明确要求，否则可能会改变结果的含义。随后，我们采用多轮迭代检查过程来分析查询的可执行性。如果 SQL

Algorithm 1: SQL 检查算法

输入： 问题 q，数据库 db，模式 S'，初始 SQL sql_{pre}

输出： 最终 SQLsql

$sql \leftarrow \text{ReduceFormat}(q, sql_{\text{pre}}, S')$
$count \leftarrow 0$
while $count <$ maxTryTime do
    $(\text{pass}, err, res) \leftarrow \text{execTool}(sql, db)$
    if pass then
        中断
    else
        $vals \leftarrow \text{valueRetrieve}(q, sql, S', db)$
        $sql \leftarrow \text{Refiner}(q, sql, S', err, res, vals)$
    end if
    $count \leftarrow count + 1$
end while

return sql

执行产生空结果、值为 0 或包含 NULL 值，则会修复查询。此修复循环将继续，直到查询成功执行或超过最大迭代次数。对于修复过程，我们根据 SQL 从数据库中检索最相关的 k 个值，并将它们与部分执行结果结合起来作为提示 LLM 进行更正的上下文。Algorithm 1详细介绍了这个过程。

4 实验

4.1 实验设置

数据集 我们在两个广泛使用的 Text2SQL 基准上评估了 GBV-SQL：Spider(Yu et al. 2018)，以评估其在超过 200 个数据库上的跨领域泛化能力（例如，飞行_2），以及更具挑战性的 BIRD(Li et al. 2023b)，以测试面对涉及噪声数据和外部知识的真实世界复杂性时的性能。虽然 Spider 2.0(Lei et al. 2024) 是一个重要的面向行业的发展，但其在长上下文理解和多工具使用上的复杂性目前阻碍了对我们语义验证机制的明确评估。我们计划在未来的工作中将 GBV-SQL 适应这样的企业场景。

基线 在我们的实验中，我们将提出的 GBV-SQL 方法与一系列基于 ICL 的方法进行了比较，包括 DIN-SQL(Pourreza and Rafiei 2023)、DAIL-SQL(Gao et al. 2024a)、MAC-SQL(Wang et al. 2025)、MAG-SQL(Xie, Wu, and Zhou 2024)、SuperSQL(Li et al. 2024a) 和 CogSQL(Yuan et al. 2025)。

评估指标 我们使用两个指标来评估我们的框架：执行准确率和有效效率分数。**执行精度**（EX）被定义为预测的 SQL 查询中，其执行结果与真实 SQL 完全匹配的比例。**有效效率得分**（VES）是由 BIRD 数据集引入的，同时考虑了生成的 SQL 的正确性和执行效率。效率得分（与执行时间相关）仅在预测和实际的 SQL 结果匹配时计算；否则，得分为 0。

实现细节 我们的实验主要以 Deepseek-v3 作为主干大型语言模型，通过 Deepseek API 访问。为了补充验证，我们还通过 OpenAI API 使用了 GPT-4o。两个模型的温度均设置为 0。在我们的 SQLChecker 模块中，最大修复迭代次数设置为 3。我们的多智能体架构借鉴了 MAC-SQL。我们的方法能够处理中等复杂度的查询，涵盖涉及单表和多表的场景。

4.2 主要结果

鸟类结果 表 1的结果显示，我们的方法显著提升了 Deepseek 基线的性能。使用 Deepseek-v3 模型，GBV-SQL 在 BIRD dev 数据集上实现了 63.23% 的 EX，比同样使用 Deepseek 模型的 MAC-SQL 提高了 5.8%，甚至超过了众多使用 GPT-4 的方法。这突显了 GBV-SQL 在 Text2SQL 任务中的有效性。

蜘蛛结果 表 2显示了我们的框架和基线方法在 Spider 数据集上的性能。GBV-SQL 在 Spider 开发集和测试集上分别实现了 79.6% 和 82.8% 的执行准确性。在测试集中，使用相同的 Deepseek-v3 模型时，我们的方法比 MAC-SQL 基线高出显著的 5.2%，达到了与依赖 GPT-4 方法相当的表现。然而，开发集上的表现出乎意料地低于预期。这种差异促使我们深入调查执行失败的本质，这揭示了问题并非源自我们的模型能力，而是我们所认定为“金标错误”：基准数据中的真理数据本身存在广泛的质量问题，详情见以下章节。

4.3 误差分析

对所有在 Spider 基准测试中执行失败的项目进行严格的手动审查表明，大多数失败并非由我们的模型引起，而是由于数据集本身的内在质量问题。因此，我们开发了一个新的框架来分类这些金标准错误。

黄金错误识别 为了确保客观和可重复分析，我们建立了一套严格协议以审核每个数据项的质量。该过程由三名精通 SQL 的研究生执行。最初，两名学生独立检查并根据我们提出的类型学（图 5）分类每个数据项中的潜在质量问题，实现了显著的标注者间一致性（Cohen's Kappa = 0.86）。第三名学生随后裁定所有分歧以确定最终分类，确保我们的发现可靠性。

方法	示例				VES
	简单	适度	挑战性的	总数	
GPT-4 (zero-shot)	–	–	–	46.35	51.75
Deepseek-v3 (zero-shot)	49.84	34.70	25.52	42.96	53.25
DIN-SQL + GPT-4	–	–	–	50.72	58.79
MAG-SQL + GPT-4	–	–	–	61.08	–
DAIL-SQL + GPT-4	63.02	45.59	43.05	54.76	–
CogSQL + GPT-4	–	–	–	59.58	64.30
CogSQL + Deepseek-v2	64.11	46.67	39.58	56.52	–
MAC-SQL + GPT-4	65.73	52.96	40.28	59.39	66.24
MAC-SQL + Deepseek-v3	62.92	50.75	43.75	57.43	68.40
GBV-SQL + Deepseek-v3 (ours)	69.51	54.62	50.69	63.23 (↑5.8%)	69.87

表 1: BIRD 开发集上的 EX 和 VES 结果。

方法	开发	测试
GPT-4 (zero-shot)	74.6	–
DIN-SQL + GPT-4	82.8	85.3
DAIL-SQL + GPT-4	84.4	86.6
CogSQL + GPT4	85.4	86.4
SuperSQL + GPT4	87.0	–
MAG-SQL + GPT-4	85.3	85.6
MAC-SQL + GPT-4	86.8	82.8
MAC-SQL + Deepseek-v3	80.1	77.6
GBV-SQL + Deepseek-v3 **(我们的)**	79.6	82.8(↑5.2%)
GBV-SQL + GPT-4o **(我们的工作)**	79.7	83.9
GBV-SQL + **无黄金错误**	96.5	97.6

表 2: EX 结果在 Spider 开发和测试数据集上的表现。相对于 MAC-SQL+Deepseek-v3 的改进（括号中显示）。“无黄金错误”标识表示在经过清理的基准子集上进行评估，从中移除了存在质量问题的条目。

黄金错误的新分类法 虽然先前的研究已经承认数据质量问题的存在，但一直缺乏明确且系统的分类。我们通过引入一种新标准来填补这一空白，该标准基于对反复出现问题的全面分析，将 Gold Errors 分为三种不同的类型：

- **类型** A **（**SQL **侧错误）**：涵盖所有源自黄金 SQL 本身的错误，假设 NLQ 是有效的。这包括语义错误，即查询未能正确或最优地表示用户的意图，以及妨碍查询执行的语法错误。
- **类型** B **（**NLQ **侧错误）**：包括所有 NLQ 本身是问题来源的情况。这包括模糊的、定义不明确的、逻辑上有误的问题，或者用给定数据库无法回答的问题。
- **类型** C **（数据库错误）**：表示数据库本身发现的错误，如缺陷的数据模式设计或不一致和“脏”的数据值。

如 5所示，每个主要类别进一步划分为细粒度的子类别层次结构。

分析 Spider **中的黄金错误** 我们对 Spider 开发集的分析揭示了大量“金标准错误”（即，真实标签中的错误），在模型失败（EX=0）的样本中发现了 183 个实例，在通过的样本（EX=1）中发现了 62 个。图 6提供了这些失败样本中各种金标准错误类别的定量分解。一个显著的例子是飞行_2 数据库中的数据污染，如图 7所示；此处，TEXT 属性中的多余空格导致许多官方的金标准 SQL 查询失败，而我们的方法能够正确处理这种“脏数据”。在手动纠正这些数据集缺陷后，我们发现只有 37 个是真正的模型故障。因此，在一个“干净”的子集上重新评估使我们的模型执行准确率在开发集上提高到 96.5%，在测试集上达到 97.6%（表 2，“无金标准错误”行）。这一发现强调了基准质量问题如何掩盖真实模型性能，这是一个普遍存在的问题，因为对 BIRD 开发集的 10%分层样本进行初步分析表明，其超过 30%的项目包含类似的金标准错误，这与近期的研究 (Shen et al. 2025; Yang et al. 2025) 相吻合。由于篇幅限制，更多关于我们 Spider 测试集分析的详细信息请参见补充材料。

4.4 消融研究

表 3展示了我们在 BIRD 开发集上的消融研究。“w/o Planner/SQLGenerator” 表示用 MAC-SQL 的模块进行替换，而其他变体则移除了指定组件。“w/o Human-like CoT” 表示将链式思维提示替换为标准的零样本提示。结果证实 GBV-SQL 框架中的每个组件都

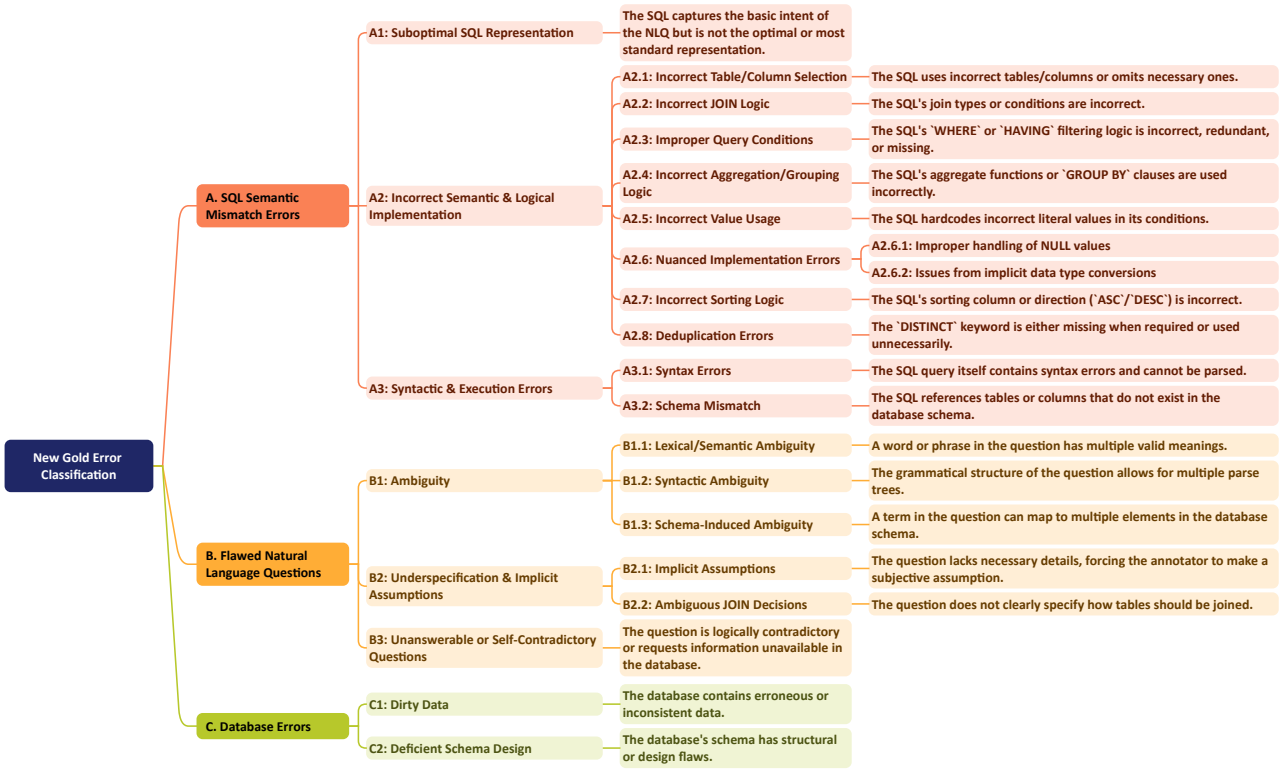

图 5: 新的黄金错误分类。

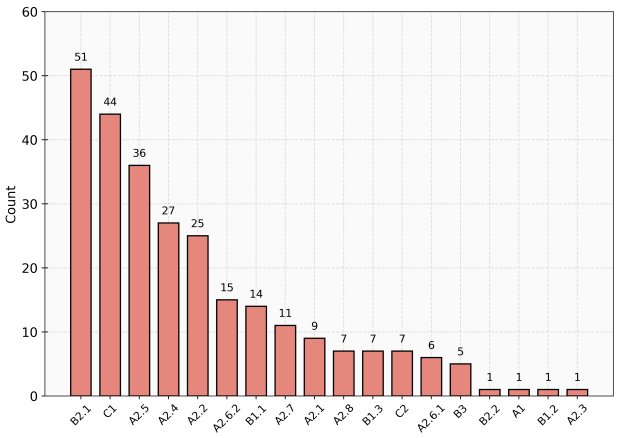

图 6: Gold Error 类型在 Spider 的开发集中的分布情况，其中 EX=0。

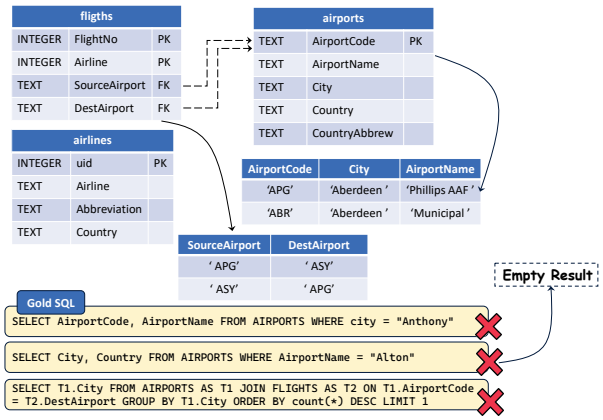

图 7: 来自 Spider 开发集中的数据库飞行_2 的一个 (C1: 脏数据) 示例。

是不可或缺的，因为移除任何一个都会降低整体性能。值得注意的是，移除处理最终格式化和可执行性分析的 SQLChecker 会导致最大的性能下降。这强调了查询可执行性和正确格式对当前基于执行（EX）评估范式的深远影响。

4.5 讨论

标准的 Text2SQL 评估将预测查询的执行结果与黄金查询进行比较，这一过程极易受到数据集质量的影响。例如，一个模糊的自然语言问题可能会使得单个黄金 SQL 查询不足以做出可靠的判断。同样地，正确的提问配以有缺陷的黄金 SQL 将仍然产生不准确的

模型	简单	模。	挑战	总数
GBV-SQL + Deepseek-v3	69.51	54.62	50.69	63.23
w/o Planner	68.76	53.98	45.83	62.13 (↓)
w/o SQLGenerator	67.78	53.76	48.61	61.73 (↓)
w/o Human-like CoT	66.70	52.90	51.39	61.08 (↓)
w/o SQL2TextValidator	68.54	52.69	47.92	61.80 (↓)
w/o SQLChecker	66.49	49.89	47.22	59.65 (↓)

表 3: GBV-SQL 在 BIRD 的开发集上的消融研究。Mod. 和 Chall. 分别代表 Moderate 和 Challenging。

EX 分数。更为严重的是，数据库级别的错误可能导致评估出现显著偏差。因此，我们强烈建议社区更加重视 Text2SQL 基准测试的完整性，采取更严格的方法进行创建和维护。基准不应被视为静态、无误的标准，而应被视为需要持续验证、修正和版本化的动态实体。

5 结论

本文研究了 Text2SQL 中的语义不一致问题，提出了一个新颖的多代理框架 GBV-SQL，该框架使用引导生成和 SQL 到文本反向翻译验证。该方法在 BIRD 基准上实现了 5.8%的准确率提升。为了更好地评估我们模型的能力，我们利用我们提出的“黄金错误”分类法对 Spider 数据集进行了分析。这一分析表明，大多数预测失败源于基准缺陷而非模型不足。在这有效子集上的重新评估将 GBV-SQL 的有效准确率提高到了 Spider 开发集和测试集的 96.5%和 97.6%，分别。因此，我们的工作提供了一个有前景的语义保真框架和一种基准整理的方法，为自动数据集验证和扩展这种验证机制到其他复杂推理任务的研究方向提出了建议。

参考文献

Bogin, B.; Berant, J.; and Gardner, M. 2019. Representing Schema Structure with Graph Neural Networks for Text-to-SQL Parsing. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 4560–4565.

Cao, Z.; Zheng, Y.; Fan, Z.; Zhang, X.; Chen, W.; and Bai, X. 2024. Rsl-sql: Robust schema linking in text-to-sql generation. arxiv:2411.00073.

Cen, J.; Liu, J.; Li, Z.; and Wang, J. 2025. SQLFixAgent: Towards Semantic-Accurate Text-to-SQL Parsing via Consistency-Enhanced Multi-Agent Collaboration. In Proceedings of the AAAI Conference on Artificial Intelligence, 49–57.

Dong, M.; Kumar, N. A.; Hu, Y.; Chauhan, A.; Hang, C.-W.; Chang, S.; Pan, L.; Lan, W.; Zhu, H.; Jiang, J.; et al. 2024. PRACTIQ: A Practical Conversational Text-to-SQL dataset with Ambiguous and Unanswerable Queries. arxiv:2410.11076.

Dong, X.; Zhang, C.; Ge, Y.; Mao, Y.; Gao, Y.; Lin, J.; Lou, D.; et al. 2023. C3: Zero-shot text-to-sql with chatgpt. arxiv:2307.07306.

Gao, D.; Wang, H.; Li, Y.; Sun, X.; Qian, Y.; Ding, B.; and Zhou, J. 2024a. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. Proceedings of the VLDB Endowment, 17(5): 1132–1145.

Gao, Y.; Liu, Y.; Li, X.; Shi, X.; Zhu, Y.; Wang, Y.; Li, S.; Li, W.; Hong, Y.; Luo, Z.; et al. 2024b. Xiyan-sql: A multi-generator ensemble framework for text-to-sql. arxiv:2411.08599.

Guo, J.; Zhan, Z.; Gao, Y.; Xiao, Y.; Lou, J.-G.; Liu, T.; and Zhang, D. 2019. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 4524–4535.

Lee, D.; Park, C.; Kim, J.; and Park, H. 2025. MCS-SQL: Leveraging Multiple Prompts and Multiple-Choice Selection For Text-to-SQL Generation. In Proceedings of the 31st International Conference on Computational Linguistics, 337–353.

Lei, F.; Chen, J.; Ye, Y.; Cao, R.; Shin, D.; Su, H.; Suo, Z.; Gao, H.; Hu, W.; Yin, P.; et al. 2024. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. arxiv:2411.07763.

Lei, W.; Wang, W.; Ma, Z.; Gan, T.; Lu, W.; Kan, M.-Y.; and Chua, T.-S. 2020. Re-examining the Role of Schema Linking in Text-to-SQL. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 6943–6954.

Li, B.; Luo, Y.; Chai, C.; Li, G.; and Tang, N. 2024a. The Dawn of Natural Language to SQL: Are We Fully Ready? Proceedings of the VLDB Endowment, 17(11): 3318–3331.

Li, H.; Zhang, J.; Li, C.; and Chen, H. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for

text-to-sql. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, 13067–13075.

Li, J.; Hui, B.; Qu, G.; Yang, J.; Li, B.; Li, B.; Wang, B.; Qin, B.; Geng, R.; Huo, N.; et al. 2023b. 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: 42330–42357.

Li, Z.; Wang, X.; Zhao, J.; Yang, S.; Du, G.; Hu, X.; Zhang, B.; Ye, Y.; Li, Z.; Zhao, R.; et al. 2024b. PET-SQL: A Prompt-Enhanced Two-Round Refinement of Text-to-SQL with Cross-consistency. arxiv:2403.09732.

Liu, X.; Shen, S.; Li, B.; Ma, P.; Jiang, R.; Zhang, Y.; Fan, J.; Li, G.; Tang, N.; and Luo, Y. 2024. A Survey of NL2SQL with Large Language Models: Where are we, and where are we going? arXiv preprint arXiv:2408.05109.

Nguyen, M. T.; Tran, K.-T.; Van Nguyen, N.; and Vu, X.-S. 2023. ViGPTQA-state-of-the-art LLMs for vietnamese question answering: system overview, core models training, and evaluations. In Proceedings of the 2023 conference on empirical methods in natural language processing: industry track, 754–764.

Pourreza, M.; Li, H.; Sun, R.; Chung, Y.; Talaei, S.; Kakkar, G. T.; Gan, Y.; Saberi, A.; Ozcan, F.; and Arik, S. O. 2024. Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql. arxiv:2410.01943.

Pourreza, M.; and Rafiei, D. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. Advances in Neural Information Processing Systems, 36: 36339–36348.

Pourreza, M.; and Rafiei, D. 2024. DTS-SQL: Decomposed Text-to-SQL with Small Large Language Models. In Findings of the Association for Computational Linguistics: EMNLP 2024, 8212–8220.

Qin, B.; Hui, B.; Wang, L.; Yang, M.; Li, J.; Li, B.; Geng, R.; Cao, R.; Sun, J.; Si, L.; et al. 2022. A survey on text-to-sql parsing: Concepts, methods, and future directions. arXiv preprint arXiv:2208.13629.

Ren, T.; Fan, Y.; He, Z.; Huang, R.; Dai, J.; Huang, C.; Jing, Y.; Zhang, K.; Yang, Y.; and Wang, X. S. 2024. Purple: Making a large language model a better sql writer. In 2024 IEEE 40th International Conference on Data Engineering (ICDE), 15–28. IEEE.

Renggli, C.; Ilyas, I. F.; and Rekatsinas, T. 2025. Fundamental Challenges in Evaluating Text2SQL Solutions and Detecting Their Limitations. arxiv:2501.18197.

Shen, J.; Wan, C.; Qiao, R.; Zou, J.; Xu, H.; Shao, Y.; Zhang, Y.; Miao, W.; and Pu, G. 2025. A Study of In-Context-Learning-Based Text-to-SQL Errors. arxiv:2501.09310.

Tai, C.-Y.; Chen, Z.; Zhang, T.; Deng, X.; and Sun, H. 2023. Exploring Chain of Thought Style Prompting for Text-to-SQL. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 5376–5393.

Wang, B.; Ren, C.; Yang, J.; Liang, X.; Bai, J.; Chai, L.; Yan, Z.; Zhang, Q.-W.; Yin, D.; Sun, X.; et al. 2025. MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. In Proceedings of the 31st International Conference on Computational Linguistics, 540–557.

Wang, Y.; and Liu, P. 2025. LinkAlign: Scalable Schema Linking for Real-World Large-Scale Multi-Database Text-to-SQL. arxiv:2503.18596.

Xie, W.; Wu, G.; and Zhou, B. 2024. Mag-sql: Multi-agent generative approach with soft schema linking and iterative sub-sql refinement for text-to-sql. arXiv:2408.07930.

Yang, Y.; Wang, Z.; Xia, Y.; Wei, Z.; Ding, H.; Piskac, R.; Chen, H.; and Li, J. 2025. Automated Validation and Fixing of Text-to-SQL Translation with Execution Consistency. Proc. ACM Manag. Data, 3(3): 134.

Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; et al. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 3911–3921.

Yuan, H.; Tang, X.; Chen, K.; Shou, L.; Chen, G.; and Li, H. 2025. Cogsql: A cognitive framework for enhancing large language models in text-to-sql translation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 39, 25778–25786.

Zelle, J. M.; and Mooney, R. J. 1996. Learning to parse database queries using inductive logic programming. In Proceedings of the National Conference on Artificial Intelligence, 1050–1055.

Zhu, X.; Wang, J.; Zhang, L.; Zhang, Y.; Huang, Y.; Gan, R.; Zhang, J.; and Yang, Y. 2023. Solving Math Word Problems via Cooperative Reasoning induced Language Models. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 4471–4485.