计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(11): 2939−2951, 2026

SCoT-SQL: 一种基于思维链引导的 Text-to-SQL 数据合成方法

任千昊[1] 刘睿珩[1,2] 张 宇[1]
[1](哈尔滨工业大学社会计算与交互机器人研究中心 哈尔滨 150001)
[2](火箭军工程大学智能与控制工程学院 西安 710025)
(qhren@ir.hit.edu.cn)

SCoT-SQL: A Chain-of-Thought Guided Approach for Text-to-SQL Data Synthesis

Ren Qianhao[1], Liu Ruiheng[1,2], and Zhang Yu[1]
[1] (*Research Center for Social Computing and Interactive Robotics*, *Harbin Institute of Technology*, *Harbin* 150001)
[2] (*School of Intelligent and Control Engineering*, *Rocket Force University of Engineering*, *Xi'an* 710025)

Abstract Traditional Text-to-SQL tasks typically require large scale annotated data for model training, which incurs significant time and labor costs. Consequently, data scarcity remains a critical bottleneck, particularly in domain specific applications. While data augmentation based on large language model (LLM) has emerged as a promising solution to this issue, most existing methods heavily rely on the distribution of original datasets and often generate synthetic data that lack reliability and interpretability. Addressing these challenges, we propose SCoT-SQL, a novel chain-of-thought guided data synthesis framework. This method effectively integrates natural language instructions with symbolic knowledge, specifically detailed database schema and table content, to guide LLM, thereby significantly reducing semantic errors during the generation process. To guarantee the high quality of synthetic data, SCoT-SQL incorporates a rigorous filtering mechanism that leverages both model self consistency voting and verification based on execution. Additionally, the framework employs a self feedback mechanism driven by chain-of-thought reasoning to calibrate synthetic data. This process not only rectifies logical inconsistencies in SQL queries but also generates explicit step by step reasoning trajectories, enhancing the interpretability of the dataset. Comprehensive experiments on the KaggleDBQA and ScienceBenchmark datasets demonstrate the efficacy of the proposed method. SCoT-SQL achieves execution accuracy improvements of 6.5% and 3% respectively, significantly outperforming previous state of the art data synthesis baselines and proving its effectiveness in low resource and complex scenarios.

Key words natural language interface of database; data synthesis; data augmentation; chain of thought; self-correction

摘 要 在 Text-to-SQL 任务中,模型通常需要大规模的标注数据进行训练,而标注这些数据往往耗费大量的时间和人力成本。因此,数据稀缺性成为制约该领域发展、尤其是在特定领域应用中的关键瓶颈。在此背景下,基于大语言模型(large language model, LLM)的数据增强虽已成为一种有效的解决方案,然而当前 Text-to-SQL 领域的大多数现有方法严重依赖于原始数据集的分布,且其合成的数据往往缺乏可靠性与可解释性。针对上述问题,提出了一种思维链引导的数据合成方法 SCoT-SQL。通过整合自然语言指令与包括数据库模式及表内容在内的符号知识,有效指引大模型减少数据合成过程中的语义错误。为确保合成数据的高质量,SCoT-SQL 结合了模型自一致性投票与执行验证机制来进行严格的数据过滤。

收稿日期: 2025-11-18; **修回日期**: 2026-05-29
基金项目: 国家自然科学基金项目(62476066)
This work was supported by the National Natural Science Foundation of China (62476066).
通信作者: 张宇(zhangyu@ir.hit.edu.cn)

此外,还引入了基于思维链推理的模型自反馈机制对合成数据进行校准,在修正 SQL 查询逻辑不一致的同时,补充了数据合成的分步推理轨迹,从而进一步增强了数据的可靠性与可解释性。在 KaggleDBQA 与 ScienceBenchmark 数据集上的实验表明,SCoT-SQL 方法的执行准确率相较于之前最先进的数据合成方法分别提升了 6.5 个百分点与 3 个百分点,证明了其在低资源及复杂场景下的有效性。

关键词 数据库自然语言接口;数据合成;数据增强;思维链;自纠错

中图法分类号 TP391.1

DOI: 10.7544/issn1000-1239.202550826 **CSTR:** 32373.14.issn1000-1239.202550826

数据库自然语言接口(natural language interface of database, NLIDB)通过将自然语言转换为结构化查询,实现了低门槛的数据库交互。Text-to-SQL 作为 NLIDB 的典型实现形式,能够自动将自然语言需求解析为结构化的 SQL 查询,在数据驱动场景中具有重要应用价值[1]。

近年来,Text-to-SQL 领域的研究在大语言模型(large language model, LLM)的推动下取得了显著进展,但仍面临两大关键挑战:数据依赖性和领域泛化性[2]。基于微调的方法需要大量高质量的标注数据,而真实应用场景中,特定领域的数据往往稀缺且标注成本高昂;同时,现有方法在跨领域场景中的泛化能力有限,尤其是在目标领域缺乏原始训练数据分布的情况下,性能往往大幅下降。

针对上述挑战,基于 LLM 的数据合成方法展现出巨大潜力,然而现有的研究主要集中在基于现有数据集的扩展或改写,虽然能在一定程度上缓解数据稀缺问题,但这类方法严重依赖原始数据分布,难以适应真实场景中无原始数据支持的情况。此外,当前的数据合成方法通常缺乏对 SQL 生成逻辑的显式建模,导致生成的数据可能存在语义不一致或逻辑错误,影响下游模型的训练效果。

为了解决这些问题,本文提出了一种基于思维链引导的 Text-to-SQL 数据合成方法(chain-of-thought guided approach for Text-to-SQL data synthesis, SCoT-SQL)。该方法基于结构化知识与 LLM 自一致性进行数据合成,在摆脱对原始数据集依赖的同时,增强了合成数据的可靠性,并通过模型自反馈方法补充数据合成的推理链,增强数据可解释性。本文主要贡献和创新点包含以下 3 个方面:

1)提出了基于结构化知识与自一致性的数据合成框架,结合数据库架构及领域内注释示例等扩展信息进行数据合成,并通过程序执行与模型自一致性投票进行验证,以确保高质量的数据合成,提升数据可靠性。

2)提出了基于思维链(chain-of-thought, CoT)的模型自反馈方法,利用思维链的推理路径对合成数据的逻辑一致性进行评估并修正,并在此基础上构建具有完整推理链的合成数据集以增强数据的质量与可解释性。

3)通过在公共数据集 KaggleDBQA 与 Science-Benchmark 上进行实验与评测,验证了 SCoT-SQL 的有效性,超越了现有基准方法。同时,本文进行了消融实验,证明了方法各模块的有效性。

1 相关工作

1.1 Text-to-SQL

将自然语言问题转化为准确的 SQL 查询(Text-to-SQL)是自然语言处理与数据库领域的重要研究方向。该领域的发展历程可追溯至早期基于规则或模板驱动的方法[3],这类方法需要大量人工干预。随着技术进步,基于 Transformer 的序列到序列模型因其在序列生成任务中的优势,逐渐成为 Text-to-SQL 的主流解决方案[4]。

代表性模型如 IRNet[5] 采用双向 LSTM 架构结合自注意力机制来编码查询语句和数据库模式。为更好地融合模式信息,后续研究提出了多种改进方案:RAT-SQL[6] 和 RASAT[7] 引入了关系感知自注意力机制;SADGA[8] 与 LGESQL[9] 则采用图神经网络建模模式-查询关系。然而,在复杂查询场景中,这些模型的性能仍明显低于人工构建的 SQL 查询质量。

随着 LLM 在自然语言处理各领域的广泛应用,Text-to-SQL 研究也迎来了新的发展机遇。早期探索主要利用 LLM 的零样本上下文学习能力生成 SQL[10]。在此基础上,DIN-SQL[11], DAIL-SQL[12], MAC-SQL[13], C3[14] 等模型通过任务分解策略进一步提升了性能。

除上下文学习外,DAIL-SQL, DTS-SQL[15], CodeS[16] 等研究尝试通过监督微调提升开源模型性能,以达到媲美或超越闭源模型的效果。而多智能体框架的

出现再次改变了 Text-to-SQL 的解决范式，研究者将复杂的 Text-to-SQL 任务分解为更简单的子任务[17]，包括模式链接、SQL 生成、SQL 优化和 SQL 选择，每个子任务都可以由基于 LLM 的智能体使用提示工程或微调技术进行管理。

与之前利用开源数据集进行监督微调或设计复杂智能体框架的研究不同，本文的重点是在标注数据稀缺的低资源场景下提升模型的 Text-to-SQL 生成能力。

1.2 Text-to-SQL 数据增强

为了解决开源数据集覆盖范围有限的问题，一些研究尝试基于现有数据集合成额外训练数据以实现数据增强[18-20]。Text-to-SQL 的数据合成涉及 2 个组件的合成：自然语言问题（NLQ）和相应的 SQL 查询。据此，以往的 Text-to-SQL 数据增强工作可以分为 3 类：1）SQL→NLQ。首先生成 SQL 查询，再根据查询内容生成对应的自然语言问题。2）NLQ→SQL。首先生成自然语言问题，再据此推导出相应的 SQL 查询。3）SQL+NLQ。同时生成匹配的 SQL 查询和自然语言问题。

表 1 根据此分类方法对最近的研究工作进行了总结，并展示了相关方法的关键步骤以及输入的上下文数据。

对于 SQL→NLQ 方法，早期基于采样 SQL 模板进行 SQL 合成，并微调编码器-解码器模型（encoder-decoder model）合成 NLQ 的解决方案逐渐被使用 SQL 模板作为上下文信息的生成式解码器（generative decoders）方法所取代。

NLQ→SQL 方法有着相同的趋势，利用问题模板的解决方案被结合上下文学习的生成解码器合成方法所取代。

此外，由于 SQL→NLQ 和 NLQ→SQL 方法的渐进式合成方案存在 SQL-NLQ 语义不对齐的现象（SQL-NLQ gap），一些研究选择 SQL+NLQ 的方法，通过联合 SQL 与 NLQ 的合成来解决这个问题。此类方法通常采用基于提示的生成方法，利用上下文学习指导 NLQ-SQL 对的合成。

不同于之前仅涉及 NLQ 与 SQL 组件的合成方法，本文在 NLQ→SQL 方法的基础上，增加了思维链

Table 1 Data Augmentation Methods for Text-to-SQL

表 1 Text-to-SQL 数据增强方法

方法（类别）	原始数据依赖	SQL 合成	NLQ 合成
文献 [2]（SQL→NLQ）	从原始数据集中提取的 PCFG	利用所构建的 PCFG 上下文数据生成 SQL 查询语句	基于原始数据微调 BART 模型并利用 SQL 数据集合成 NLQ
文献 [21]（SQL→NLQ）	从原始数据集中提取的 SQL 模板分布	通过构建抽象语法树语法并持续生成 SQL 语句，直至覆盖原始数据集中 80% 的 SQL 模板	将 SQL 语句分解表示为多个片段，并训练基于复制机制的 seq2seq 模型来实现 SQL→NLQ，最后组合输出片段
文献 [22]（SQL→IR→NLQ）	从原始数据集中提取的 SQL 模板	基于从原始数据集中提取 SQL 模采用架构加权的列抽样方法进行模板填充	微调 T5/Large 模型来实现从 SQL→IR→NLQ 的完整转换流程
方法（类别）	原始数据依赖	NLQ 合成	SQL 合成
H-NEURSYN[23]（ES→NLQ→SQL）	从原始数据中提取的实体序列（ES）	训练基于 T5 的 Database-to-ES 和 ES-to-NLQ 模型，并对数据库采样生成实体序列，进而合成 NLQ 数据集	微调 T5 模型增强 NLQ→SQL 能力，并结合执行验证与去重机制使用 NLQ 数据合成 SQL 数据集
SymGen[24]（NLQ→SQL）	从原始数据集随机抽取的 NLQ 与 SQL 数据	基于目标数据库架构以及从原始数据集中随机采样的 10 个 NLQ 示例，采用 Codex/davinci-002 模型通过提示工程合成 NLQ 数据	基于目标数据库架构以及从原始数据集中随机采样的 10 个 SQL 示例，采用 Codex/davinci-002 模型通过提示工程合成 SQL 数据
文献 [25]（NLQ→FACTS→SQL）	原始数据集的 NLQ 模板	通过预定义模板合成 NLQ，并借助 CodeX 模型将 NLQ 为多个事实要素的组合	微调 T5/Large 模型增强 FACTS→SQL 能力，并采用提示方法结合生成的 FACTS 数据，实现 FACTS→SQL 转换
方法（类别）	原始数据依赖	NLQ+SQL 联合合成	
文献 [26]（NLQ+SQL）	从原始数据集中提取的 SCFG	通过筛选 90 个高频 SQL 模板程序，为每个 SQL 模板匹配 4 个关联的自然语言查询（NLQ）模板，并在给定数据库架构下基于提取的随机上下文无关语法（SCFG）规则采样填充这些模板对，从而构建 NLQ+SQL 数据集	
文献 [27]（T→SQL+NLQ）	从原始数据集中提取 SQL 模板	基于数据库架构信息和已提取的 SQL 模板，通过 GPT-4 生成特定于该数据库的详细描述，最后利用这些上下文数据（包括 SQL 模板、数据库架构和生成的数据库描述）提示 GPT-4 合成匹配的 SQL 查询及其对应的 NLQ	
方法（类别）	原始数据依赖	NLQ 合成	SQL 合成	CoT 合成
本文方法（NLQ→SQL→CoT）	不依赖于原始数据集	基于数据库架构（包含主外键关系）、各表随机抽样的 3 行数据记录以及 3 个经过人工标注的示范样例，采用 Llama-3-8b 模型通过提示工程生成合成数据	基于先前合成的 NLQ、包含表格内容采样的数据库架构以及 3 个人工标注的示例，通过 Llama-3-8b 模型合成 SQL，并进行自一致性与执行过滤	基于 NLQ 与 SQL 数据、数据库架构以及 3 个示例，通过 Llama-3-8b 模型对 SQL 优化并生成包含 CoT 的中间过程

增强模块，在对 SQL 进行模型自反馈与优化的同时，还增强了数据的可解释性。

2 SCoT-SQL 数据合成方法

如图 1 所示，本文的数据合成框架采用渐进式合成方案，包括 3 个关键步骤：结构化知识引导的问题合成、集成自一致性过滤的 SQL 合成和基于思维链方法的模型自反馈。

该方案首先利用数据库架构及其字段描述等结构化知识，引导模型生成与数据库语义一致的自然语言问题，接下来结合模型自一致性投票以及执行过滤等方法生成并筛选 SQL 语句，从而提高合成数据的质量，最后再引入思维链方法进行模型自反馈，对合成数据中的 SQL 进行优化，并给出生成 SQL 查询的分步推理过程。

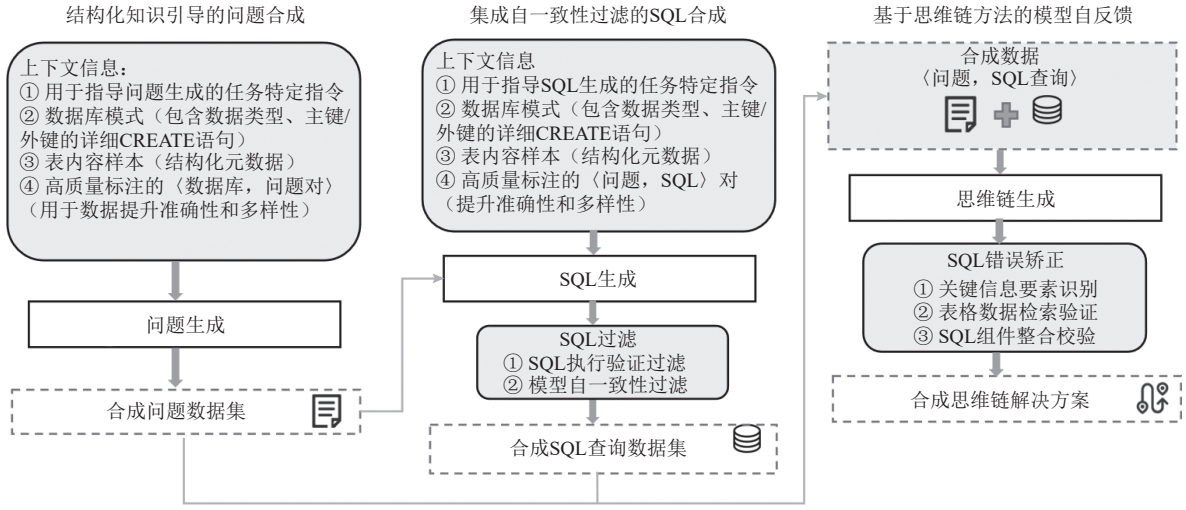


Fig. 1 Overall architecture of SCoT-SQL method

图 1 SCoT-SQL 方法总体架构

2.1 结构化知识引导的问题合成

结构化知识引导的问题合成依托大模型的上下文学习能力实现。类似于人类注释者需要依据注释规范完成标注任务，LLM 的标注性能同样高度依赖于提示设计。为此，本文在数据合成模块提示构建中整合了 3 类关键信息：1)自然语言指令；2)任务相关的领域结构化知识(包括数据库架构信息和表格内容)；3)少量标注示例。

对于数据库架构信息，本文使用基于丰富模式表征的底层数据库表示方法，其格式以一系列 CREATE 语句呈现，该表征不仅详细定义了各列的数据类型，还明确标注了主键与外键之间的关联关系。Ye 等人[24]的研究发现，这种细粒度的模式表征能显著减少数据合成中的语义错误，例如避免对文本字段错误应用算术函数，同时确保表连接基于正确的键关系。此结果与 Hu 等人[22]的研究一致，表明明确的模式语义信息是提升文本到 SQL 转换准确性的关键因素。

同时，本研究通过随机采样策略从数据库各表中提取具有代表性的数据样本，将其作为结构化元数据与数据库模式信息进行整合，从而在数据合成任务中构建更具信息量的上下文表征。为优化数据合成模块的性能表现，本研究在上下文信息中引入少量标注样本作为示例，旨在提升模型合成数据的准确性与多样性，具体提示词组织格式如图 2 所示。

```
Input
1. Schema: {CREATE_STATEMENTS}
(data types, primary/foreign keys)
2. Sample Data: {TABLE_SAMPLES}
3. Few-shot Examples: {EXAMPLES}

Task
Generate NL questions that:
• Require aggregation, filtering, or joins
• Align with database semantics
• Use correct join key relationships

Output
Natural language question
```

Fig. 2 Data synthesis prompts

图 2 数据合成提示词

2.2 集成自一致性过滤的 SQL 合成

基于问题合成模块的生成结果，本文进一步提出了一种集成自一致性过滤机制的 SQL 查询合成方法，图 3 是该模块流程的一个示例。

针对 LLM 在 SQL 合成过程中可能违反 SQL 语法规范的问题，本文在沿用问题合成方法进行 SQL

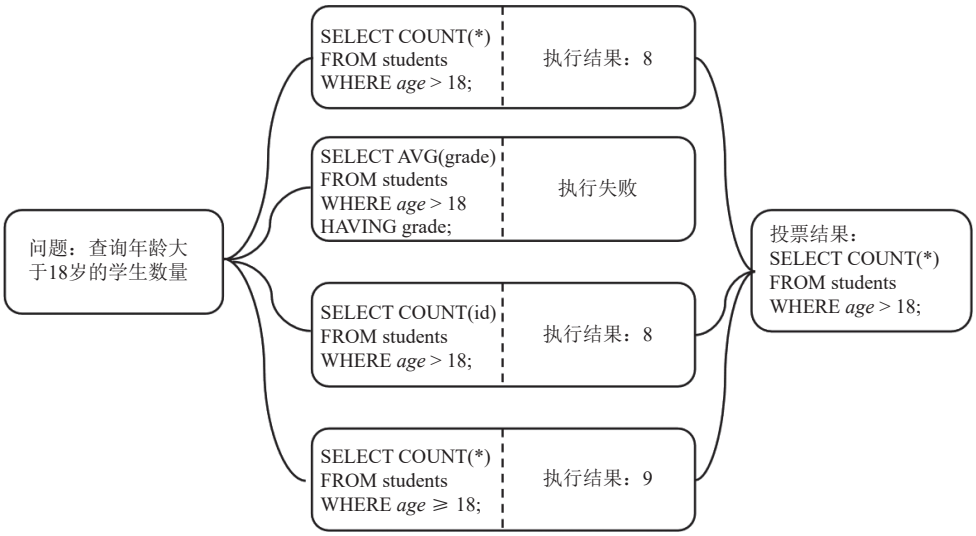


Fig. 3 SQL synthesis examples

图 3 SQL 合成示例

合成的同时，还采用束搜索（beam search）策略并行生成多个候选查询语句，以提高有效输出的覆盖范围。在此基础上，本文引入基于自一致性的数据过滤方法，对包含多个候选答案的合成数据集C进行进一步验证。

具体而言，本文首先对合成数据集C的所有候选答案c_i执行严格的语法验证，利用数据库执行检测函数$exec(\cdot)$来自动过滤存在语法错误的无效候选，保留可执行的候选答案集合作为后续处理的输入。式（1）和式（2）分别给出了语法过滤机制以及语法验证函数的定义。

$$C' = \{c_i \in C \mid exec(c_i) \neq \perp\} 。 \tag{1}$$

$$exec(c_i) = \begin{cases} y_i, & \text{if 可执行} 。 \\ \perp, & \text{其他} 。 \end{cases} \tag{2}$$

本文采用精确匹配（exact match, EM）作为相似度度量方法，其输出值域为{0, 1}的二元空间。通过计算各候选答案执行结果间的相似度矩阵$\boldsymbol{S}$，构建基于阈值筛选的置信度评估机制，相似度矩阵中的元素定义如式（3）所示。

$$S_{ij} = \mathbb{I}_{y_i \equiv y_j} = \begin{cases} 1, & y_i = y_j , \\ 0, & \text{其他} 。 \end{cases} \tag{3}$$

如式（4）所示，基于相似度矩阵的元素S_{ij}可以计算得到相似度分数$\bar{s}_i$：

$$\bar{s}_i = \frac{1}{|C'|-1} \sum_{j \neq i} S_{ij} 。 \tag{4}$$

通过仅保留相似度分数$\bar{s}_i$超过预设阈值τ的候选答案c_i，可以确保最终输出结果的可靠性，其过程如式（5）所示。

$$C'' = \{c_i \in C' \mid \bar{s}_i \geqslant \tau\} 。 \tag{5}$$

如式（6）所示，对于经过相似度阈值过滤的数据集合C''，本文将选择在生成过程中具有最大生成概率p的答案c^*。

$$c^* = \arg\max_{c_i \in C''} (\ln p(c_i)) 。 \tag{6}$$

2.3 基于思维链方法的模型自反馈

思维链，一种引导大模型逐步思考解决问题的方式，在复杂推理任务中已被证明具有良好的效果。本文借助思维链方法对合成数据进行增强，所使用的上下文数据包括自然语言指令、数据库架构信息以及前文合成的自然语言问题和 SQL 查询。

典型的思维链方法采用分步推理机制实现 SQL 查询的生成与优化[28-29]。该方法首先通过语义解析识别问题中的关键信息要素，据此确定需要检索的相关数据表及列筛选条件，之后基于查询需求智能整合表连接、条件筛选、聚合运算和分组操作等 SQL 组件，通过迭代式构建过程最终生成完整查询，本文所使用的思维链提示词格式如图 4 所示。

在实现层面，本研究采用与前文合成数据阶段相同的预训练语言模型，结合思维链方法与模型自反馈机制对生成的 SQL 查询进行优化。实验发现，初始合成的 SQL 查询存在若干可优化项，包括冗余列选择和非最优连接路径等问题。得益于思维链框架的分步推理特性，语言模型能够在中间推理过程中有效识别并修正这类问题，从而生成结构更合理、语义更精确的 SQL 查询。这一发现与 Madaan 等人[30]的研究结论相印证，该研究同样表明了语言模型在检测和修正 SQL 语法细微错误方面具有较强的能力。

```
Role
Senior data analyst specializing in SQL.
Task: Translate NL question to SQL with detailed reasoning trace.

Input
1. Database Schema: {CREATE_STATEMENTS}
2. Question: {NL_QUESTION}
3. Candidate SQL: {SYNTHESIZED_SQL}

Review points
The candidate SQL may contain:
① Unnecessary table/column selections
② Incorrect or excessive JOINs
③ Misalignment with the question
④ Opportunities for simplification

CoT reasoning steps
① Semantic parsing: identify entities/filters
② Schema linking: map to tables/columns
③ Query construction: build SQL clauses
④ Self-review: check and optimize query

Output format
Thought: [Step-by-step reasoning process]
```sql
SELECT ··· FROM ··· WHERE ···
```
```

Fig. 4 CoT synthesis prompts

图 4 思维链合成提示词



Fig. 5 Model self-feedback examples

图 5 模型自反馈示例

因此,引入思维链方法不仅能够生成可解释的推理过程,还能显著提升合成数据的准确性与可用性。图 5 是模型自反馈模块的一个示例。

3 SCoT-SQL 合成数据集

本文基于第 2 节所述的合成方法,在 2 个低资源复杂 Text-to-SQL 基准测试集 KaggleDBQA[31] 和 ScienceBenchmark[32] 上构建了合成数据集,图 6 展示了原始数据与合成数据的嵌入空间分布对比。

实验选用这 2 个基准主要基于以下考量:首先,它们代表了真实场景中的复杂查询需求;其次,作为低资源数据集,它们更能验证本文方法在数据稀缺情况下的有效性。

在表 2 中,本文对所使用的 2 个基准及其子数据集划分的数据库数量、表数量、列数量、行数量、每个表的行数、数据库大小、问答对个数作了详细展示。

3.1 SCoT-SQL-KaggleDBQA

KaggleDBQA 是一个规模相对较小的数据集,其设计初衷在于填补学术基准(如 Spider[33])与实际工业数据库之间的鸿沟。该数据集直接由 Kaggle 平台采集的原始工业数据库构建而成,仅经过最小化预处理以保留真实场景的数据特征。与 Spider 等学术基准相比,KaggleDBQA 虽然数据库模式较为简单,但包含了大量源自实际业务场景的复杂 SQL 查询,同时完整保留了工业环境中典型的非标准化命名惯例(如缩写列名、领域特定分类值等)和遗留系统迁移带来的数据特征,为研究文本到 SQL 在真实工业场景中的应用提供了独特的测试平台。

本文基于 KaggleDBQA 的数据库架构合成了 5 000 条带有思维链推理过程的 NLQ-SQL 对,并进一步分析了这些数据在 NLQ 与 SQL 空间分布上的特征。

如图 6(a)所示的自然语言问题嵌入空间 t-SNE 可视化中,合成数据点形成了数十个紧密的语义集群。值得注意的是,尽管本文的问题合成过程完全独立于原始训练数据,但这些合成数据集群在空间分布上呈现出围绕原始问题聚集的特征模式。这种现象清晰地表明,本文提出的合成方法能够有效地捕捉原始数据的问题分布特性,在语义空间层面实现了与真实数据分布的高度一致性。

图 7 中合成数据与原始数据的 SQL 复杂度对比分析显示,在 KaggleDBQA 数据集上,本文方法生成的查询具有复杂度提升的特征。而图 6(b)展示了 SQL 数据 t-SNE 可视化结果,合成数据不仅覆盖了原始查询大部分的分布区域,还拓展了原始数据未能触及的多个语义空间区域。

这一空间分布特征明确表明,本文的合成方法

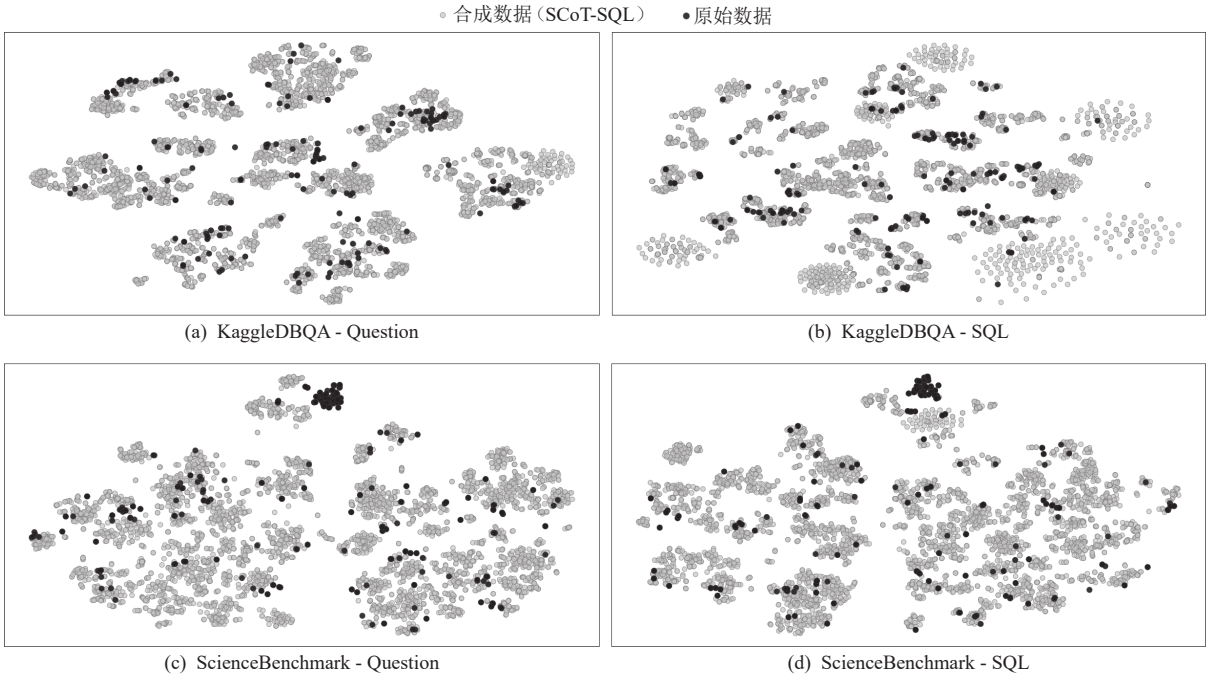


Fig. 6 Embedding space of KaggleDBQA data and ScienceBenchmark data

图 6 KaggleDBQA 数据与 ScienceBenchmark 数据嵌入空间

Table 2 Statistics of KaggleDBQA and ScienceBenchmark

表 2 KaggleDBQA 与 ScienceBenchmark 统计信息

KaggleDBQA 及其子数据集划分							
数据库	数据库数	表数	列数	行数	每个表的行数	大小/GB	问答对数
KaggleDBQA	8	17	179	4.7×10^6	280×10^3	0.4	272
GND	1	1	14	1×10^3	1×10^3	0.001	22
GMC	1	1	6	5×10^3	5×10^3	0.001	18
PC	1	2	34	2.3×10^6	1.2×10^6	0.12	34
SMS	1	4	17	14×10^3	4×10^3	0.001	19
THB	1	5	44	61×10^3	12×10^3	0.006	27
USW	1	1	19	1.9×10^6	1.9×10^6	0.196	25
WCH	1	2	10	237×10^3	119×10^3	0.01	28
WSD	1	2	37	217×10^3	108×10^3	0.034	12
ScienceBenchmark 及其子数据集划分							
数据库	数据库数	表数	列数	行数	每个表的行数	大小/GB	问答对数
ScienceBenchmark	3	50	249	151×10^6	3×10^6	19.1	5 032
CORDIS	1	19	82	671×10^3	35×10^3	1	200
OncoMX	1	25	106	65×10^6	2.6×10^6	12	199
SDSS	1	6	61	86×10^6	14×10^6	6.1	200

具有生成多复杂度层级 SQL 查询的能力，特别是相比人工标注的原始数据，能够产生更具挑战性的复杂查询实例。

3.2 SCoT-SQL-ScienceBenchmark

ScienceBenchmark 是一个专为评估自然语言到 SQL 系统而设计的高复杂度基准测试，其数据库规模、结构复杂性和领域多样性远超 Spider 等传统学术基准。该基准包含 3 个高度专业化的科学领域数据库，其中最具挑战性的子数据库包含 25 个表、6 500 万条数据，大小达 12 GB，远超 Spider 的平均水平（3.5 个表、8 000 条数据、0.03 GB）。

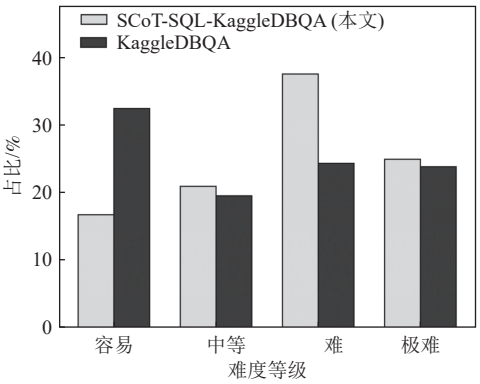


Fig. 7 Difficulty comparison between KaggleDBQA and the corresponding synthetic dataset

图 7 KaggleDBQA 与对应合成数据集难度对比

此外，ScienceBenchmark 提供了规模较大的描述性文本，其中每个项目的平均描述长度达到 1 821 个字符，涉及的主题范围广泛，涵盖从信息科学到核裂变等高度专业化的领域。这种规模和复杂性使得 ScienceBenchmark 成为当前最具挑战性的 Text-to-SQL 基准之一，尤其适合评估模型在真实工业级数据库

上的表现。

本文基于 ScienceBenchmark 数据库架构，构建了包含 4 432 条带有思维链推理过程的 NLQ-SQL 对数据集。如图 6 所示，通过对比合成数据与原始数据在嵌入空间中的分布特征，可以发现本研究生成的合成数据覆盖了原始数据空间的绝大部分语义空间区域。

除此之外，原始数据呈现出明显的聚集性分布特征，大量样本密集堆积在特定区域，表明其语义覆盖存在局限性。相比之下，本文生成的合成数据在低维空间中展现出更为均匀的分布特性。这体现本文的合成数据能够有效模拟真实数据中的语义多样性，不存在明显的语义缺失或偏差。

图 8 中合成数据与原始数据 SQL 难度对比显示，在 SQL 生成难度维度上，本文合成数据与人工标注数据的难度分布匹配度达到 94.1%，表明二者在任务复杂度层面具有高度一致性。

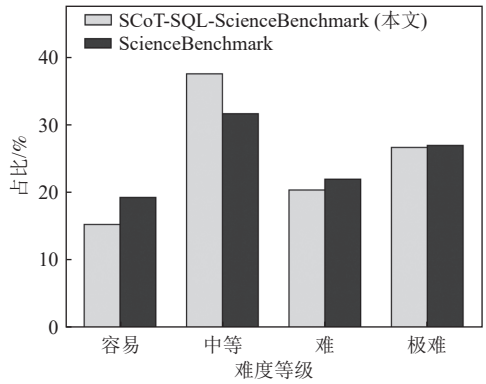


Fig. 8 Difficulty comparison between ScienceBenchmark and the corresponding synthetic dataset

图 8 ScienceBenchmark 与对应合成数据集难度对比

4 实验与分析

实验细节部分，本文数据合成与模型微调阶段均采用 Llama-3-8B-Instruct[34] 作为基座模型，该模型是基于大规模多领域语料进行预训练并经过指令微调的高性能语言模型。在数据过滤阶段，本文采用的相似度阈值 τ 为 0.25。在模型微调阶段，本文采用低秩自适应方法(LoRA)，将秩维度(rank)设为 64，缩放系数(alpha)设为 32，并将可训练适配器注入模型的 query(q_proj)和 value(v_proj)投影层，同时保持基础模型的原始参数冻结。本文采用参数 β_1=0.9，β_2=0.999，ϵ = 1E−8的 AdamW 优化器来优化训练目标，学习率为 2E−4，累积梯度的批量大小设置为 16，并在本文合成的 2 个数据集上训练了 8 轮。

评价指标部分，与之前的 Text-to-SQL 任务类似，本文采取准确率(*EX*)作为实验的评价指标。准确率通过比较模型生成的 SQL 查询与真实 SQL 查询在数据库上的执行结果是否一致来评估模型的性能。具体而言，给定一个测试样本，若模型生成的 SQL 查询执行后得到的结果与真实 SQL 查询的执行结果完全相同，则认为该样本预测正确。最终，执行准确率计算为预测正确的样本数(*E*)占总测试样本数(*N*)的比例，其计算公式如式(7)所示。

$$EX = \frac{E}{N}。 \tag{7}$$

4.1 对比方法与基线设置

4.1.1 直接推理类基线

本文对 SCoT-SQL 方法与多种 LLM 在 KaggleDBQA 与 ScienceBenchmark 上的评测结果进行了系统性对比分析，涵盖闭源与开源两大类别，表 3 与表 4 分别展示了 2 种数据集下不同模型的对比结果。

在闭源模型方面，本文选取了 GPT-3.5[35] 和 Codex/davinci-002[36] 作为对比模型；在开源模型方面，则系统考察了 Deepseek-Coder[37], Qwen2[38], Meta-Llama-3[34], Code Llama[39] 等主流模型，以及 DeepSeek-R1[40] 系列的多个蒸馏变体。

这些对比模型的参数量均与本研究所用基座模型的参数量相当或更大，且具有广泛的代表性：从模型性质看，既包含闭源商业模型也包含开源模型；从规模维度看，覆盖了从中小规模到超大规模的不同级别；从专业领域看，既包含通用语言模型也包含代码专用模型；从架构设计看，既包含传统密集架构也包含混合专家(MoE)等创新设计。

4.1.2 数据合成类基线

在数据合成方法对比上，我们选取了 3 种代表不同优化路径的数据合成方案，从多维度验证本文方法的优势。针对合成数据的正确性验证问题，我们对比了 SymGen[24]。SymGen 代表了“基于一致性过滤”的技术路线，通过利用执行结果的一致性来剔除 LLM 生成过程中的噪声，在小样本场景下具有很强的竞争力。针对合成语料与真实语料的句法结构差异问题，我们引入了 Niu 等人[25] 提出的 SNQSD 方法。该工作主张通过句子分解(sentence decomposition)技术来规范化自然语言输入的结构，旨在解决合成数据在复杂句式上泛化能力不足的问题，这为本文提供了一个结构维度的对比视角。

此外，我们还引入了近期提出的 OmniSQL[41] 作

Table 3 Experimental Results on KaggleDBQA and Its 8 Sub-Databases Test Sets

表 3 KaggleDBQA 及其 8 个子数据库测试集实验结果

通用模型（7B~175B）										
模型	数据量	准确率	GND	GMC	PC	SMS	THB	USW	WCH	WSD
Meta-Llama-3-8B-Instruct	Zero Shot	0.249	0.409	0.389	0.294	0.105	0.037	0.23	0.286	0.25
	3 Shot	0.297	0.5	0.5	0.265	0.105	0.037	0.36	0.286	0.5
DeepSeek-R1-Distill-Llama-8B	Zero Shot	0.227	0.409	0.333	0.088	0.158	0.222	0.2	0.214	0.333
	3 Shot	0.184	0.364	0.222	0	0.105	0.222	0.24	0.214	0.167
DeepSeek-R1-Distill-Qwen-14B	Zero Shot	0.308	0.455	0.333	0.353	0.263	0.148	0.32	0.214	0.5
	3 Shot	0.351	0.545	0.389	0.235	0.263	0.111	0.52	0.429	0.417
Qwen2-7B-Instruct	Zero Shot	0.308	0.591	0.444	0.235	0.158	0.111	0.4	0.143	**0.667**
	3 Shot	0.292	0.545	0.444	0.235	0.158	0.185	0.32	0.107	0.583
Meta-Llama-3-70B-Instruct	Zero Shot	0.308	0.409	0.444	0.147	0.316	0.259	0.4	0.214	0.5
	3 Shot	0.416	**0.636**	**0.611**	0.147	**0.474**	0.185	0.56	0.429	0.583
Codex/davinci-002	Zero Shot	0.268	-	-	-	-	-	-	-	-
GPT-3.5	Zero Shot	0.243	-	-	-	-	-	-	-	-
代码模型（6.7B~13B）										
模型	数据量	准确率	GND	GMC	PC	SMS	THB	USW	WCH	WSD
CodeS-7B	Zero Shot	0.097	0.136	0.167	0	0.053	0.148	0.24	0.036	0
	3 Shot	0.357	0.5	0.5	0.353	0.158	0.296	0.52	0.214	0.333
CodeLlama-13B-Instruct-hf	Zero Shot	0.146	0.227	0.278	0.118	0	0.111	0.24	0.107	0.083
	3 Shot	0.141	0.227	0.222	0.059	0	0.074	0.24	0.143	0.25
DeepSeek-Coder-6.7B-Instruct	Zero Shot	0.189	0.318	0.222	0.176	0.105	0.111	0.32	0.107	0.167
	3 Shot	0.256	0.409	0.389	0.324	0.053	0.074	0.4	0.143	0.333
基于数据合成的方法（以 Meta-Llama-3-8B-Instruct 为基座模型）										
模型	数据量	准确率	GND	GMC	PC	SMS	THB	USW	WCH	WSD
OmniSQL-Gen	5 000	0.319	0.545	0.444	0.294	0.105	0.185	0.36	0.25	0.5
SymGen	5 000	0.373	0.364	0.5	0.265	0.211	0.222	**0.64**	0.429	0.417
SNQSD	5 000	0.373	0.5	0.5	0.235	0.158	0.259	0.56	**0.464**	0.333
本文	5 000	**0.438**	**0.636**	0.556	**0.471**	0.316	0.148	**0.64**	0.286	0.583

注：“-”代表只有数据集总和结果，没有子数据集结果。黑体数值表示最优结果。

为另一类数据合成基线。与前述方法通常依赖给定数据库 Schema 生成不同，OmniSQL 采用了一种从零构建数据库与查询的全合成策略（synthesis from scratch），并遵循 SQL-to-NLQ 的反向生成路径，即先构建复杂的 SQL 查询逻辑，再利用大模型将其回译为自然语言问题。为了公平评估不同合成策略在同等数据资源下的效能，我们在实验中从 OmniSQL 开源的高质量数据集中抽取了与本文方法同等规模的数据子集对基座模型进行微调。将 SCoT-SQL 与这种代表性的“全链路合成”基线进行对比，旨在验证在数据量受控的前提下，本文针对正向推理路径（NLQ-to-SQL）所设计的结构感知及自一致性过滤机制以及在提升数据质量方面的有效性。

4.2 SCoT-SQL-KaggleDBQA 实验结果

本文在 KaggleDBQA 基准的 8 个子数据库测试集上进行了全面评估，实验结果表明本文方法显著提升了模型在 Text-to-SQL 任务的性能表现。其结果如表 3 所示，表 3 中 Codex 与 GPT-3.5 结果分别来自于 Baumgartner 等人[27] 和 Wang 等人[42] 的报告。

本文方法在完整数据集的准确率上超越了与其同规模或规模更大的一众对比模型，并在最多的子数据集划分上达到了最佳结果。具体而言，本文方法在测试集上达到了 0.438 的准确率，相比基座模型的 Zero Shot（0.249）和 3 Shot（0.297）设置分别实现了 18.9 个百分点和 14.1 个百分点的性能提升。特别值得关注的是，尽管采用更小规模的模型参数，本文方

法仍以 2.2 个百分点的优势超越了同系列 Meta-Llama-3-70B-Instruct 模型的 3 Shot 结果。与其他数据合成方法对比,在同等数据量前提下,本文方法实现了最先进的性能。并且,虽然本文方法仅使用 5 000 条合成数据进行 LoRA 微调,但其效果超越了在大规模 Text-to-SQL 数据(45 万条)上进行后训练的 CodeS-7B。

4.3 SCoT-SQL-ScienceBenchmark 实验结果

本文在 ScienceBenchmark 基准上进行了全面的评估,并报告了本文方法整体准确率以及在 3 个子数据库(CORDIS, OncoMX, SDSS)上的执行准确率。本文方法在专业科学领域复杂基准测试中展现出卓越的领域泛化能力和文本理解性能,结果如表 4 所示。

Table 4 Experimental Results on ScienceBenchmark and Its Three Sub-Databases Test Sets

表 4 ScienceBenchmark 及其 3 个子数据库测试集的实验结果

通用模型(8B~70B)					
模型	数据量	准确率	CS	OX	SS
Meta-Llama-3-8B-Instruct	1 Shot	0.214	0.28	0.343	0.02
DeepSeek-R1-Distill-Llama-8B	1 Shot	0.281	0.46	0.363	0.02
CodeLlama-13B-Instruct-hf	1 Shot	0.251	0.34	0.39	0.02
Meta-Llama-3-70B-Instruct	1 Shot	**0.352**	**0.53**	0.475	0.05
代码模型(7B~13B)					
模型	数据量	准确率	CS	OX	SS
CodeS-7B	1 Shot	0.216	0.33	0.313	0.06
CodeLlama-13B-Instruct-hf	1 Shot	0.251	0.34	0.39	0.02
基于数据合成的方法(以 Meta-Llama-3-8B-Instruct 为基座模型)					
模型	数据量	准确率	CS	OX	SS
SymGen	4 432	0.322	0.43	0.505	0.03
SNQSD	4 432	0.308	0.48	0.434	0.01
Syn-Science	4 423	0.207	0.31	0.273	0.04
本文	4 423	**0.352**	0.44	**0.525**	**0.09**

注:CS,OX,SS 为 CORDIS,OncoMX,SDSS 缩写。黑体数值表示最优结果。

相较于基座模型准确率 0.214 的结果,本文实现了 13.8 个百分点的提升。与主流通用模型及经过大规模领域适应训练的专用模型相比,本文方法在同等规模下保持领先优势,并与参数更大的 Meta-Llama-3-70B-Instruct 模型保持相当的准确性。此外,在合成相同数据量的前提下,本文方法相较于最先进的数据合成方法 SymGen 提升了 3 个百分点。

进一步分析子数据集的结果,本文方法在 OncoMX 与 SDSS 这 2 个规模更大且更为复杂的子集的准确率超越了所有对比方法,表明其应对复杂表间关系时的优越性。这一结果与前文对合成数据集难度分析相印证,本文的数据合成方法具有生成多复杂度层级 SQL 查询的能力,相较于对比方法能够合成更具挑战性的复杂查询实例。

4.4 消融实验结果

为了更加明确本文提出的机制对模型的影响,本节进行了消融实验,分别探究数据合成过滤模块与模型自反馈模块对结果的影响。

4.4.1 对数据合成过滤模块的消融

为了评估数据合成过滤模块对结果的影响,本文通过移除合成数据中的 SQL 部分,仅使用 NLQ 数据合成思维链,对 Meta-Llama-3-8B-Instruct 进行微调,从而开展消融实验。具体而言,在思维链增强模块,该部分不再提供合成 SQL 作为推理过程合成的扩展信息,也不再对合成 SQL 进行优化,而是直接针对 NLQ 同时合成 CoT 与 SQL,并使用这部分数据对模型进行微调,消融结果如表 5 所示,删除 SQL 数据后导致模型在 KaggleDBQA 上准确率下降 3.3 个百分点,在 ScienceBenchmark 上下降 2.4 个百分点。这一结果充分说明数据合成过滤模块的有效性。

Table 5 Ablation Study Results

表 5 消融实验结果

方法	准确率	
	KaggleDBQA	ScienceBenchmark
SCoT-SQL(完整方法)	0.438	0.352
去除数据合成过滤模块	0.405	0.328
去除模型自反馈模块	0.373	0.321

4.4.2 对模型自反馈模块的消融

本节进一步探究了模型自反馈模块对整体方法的影响。为了证明模型自反馈模块的有效性,本文使用数据合成过滤模块得到的合成数据集微调 Meta-Llama-3-8B-Instruct。如表 5 所示,移除基于思维链的模型自反馈模块后,模型准确率在 KaggleDBQA 与 ScienceBenchmark 上分别下降 6.5 个百分点与 3.1 个百分点。本文分析认为,这一结果一方面源于为合成数据补充推理过程有助于提升模型的推理能力,另一方面也得益于思维链推理过程中对错误 SQL 查询的纠正从而提升了数据整体质量。

4.4.3 参数敏感性分析

为了探究自一致性阈值 τ 对数据质量的具体影响,本节在 KaggleDBQA 数据集上进行了参数敏感性分析实验,结果如图 9 所示。

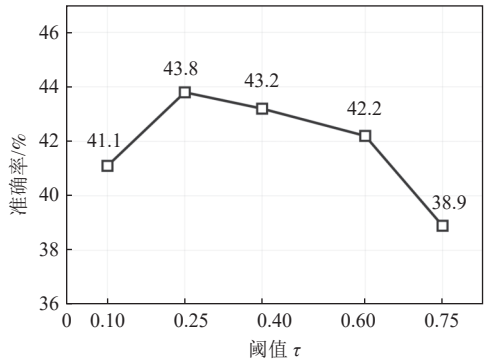


Fig. 9 Impact of different thresholds on accuracy of KaggleDBQA

图 9 不同阈值对 KaggleDBQA 准确率的影响

随着τ值的增加，模型在 KaggleDBQA 上的执行准确率呈现出倒 U 型趋势。这一结果揭示了数据筛选中噪声控制与信息增益之间的博弈关系：当阈值设置过低（τ=0.1）时，训练集中混入了大量执行结果偶然一致但逻辑错误的噪声样本，误导了模型的梯度下降方向，导致性能受损；反之，当阈值设置过高时，尽管保证了数据的准确性，模型性能却不升反降。深入分析表明，满足极高一致性的样本多为模型已完全掌握的简单查询，其信息增益极低。仅在此类简单样本上训练，模型难以习得处理复杂长尾问题的能力。相比之下，τ=0.25 时能有效地在过滤噪声的同时，保留了处于模型“认知边界”的高价值困难样本，从而在保证数据可靠性的基础上最大化了微调收益。

5 结　论

本文提出了 SCoT-SQL 数据合成框架，以结构化知识指引、思维链生成与自一致性过滤机制为核心，在低资源条件下实现了高质量、可控且可解释的 NLQ-SQL 数据构建。实验结果表明，通过 SCoT-SQL 方法合成的训练数据能够有效提升下游模型的性能，在 2 个代表性的基准 KaggleDBQA 和 ScienceBenchmark 上相较于之前最先进的数据合成方法分别提升了 6.5 个百分点与 3 个百分点。消融分析进一步验证了思维链与过滤模块对合成数据质量的关键贡献。在未来工作中，我们将进一步探索如何将本文方法拓展到更丰富的现实场景中，包括支持更多的 SQL 方言（如 PostgreSQL 等）以及多轮 Text-to-SQL 交互场景。

作者贡献声明：任千昊负责方案整体设计并撰写论文；刘睿珩负责部分算法思路和实验方法；张宇提出指导意见并修改论文。

参　考　文　献

[1] Pan Xuan, Xu Sihan, Cai Xiangrui, et al. Survey on deep learning based natural language interface to database[J]. Journal of Computer Research and Development, 2021, 58 (9): 1925−1950 (in Chinese)
(潘璇, 徐思涵, 蔡祥睿, 等. 基于深度学习的数据库自然语言接口综述[J]. 计算机研究与发展, 2021, 58(9): 1925−1950)

[2] Wang Bailin, Yin Wenpeng, Lin Xi Victoria, et al. Learning to synthesize data for semantic parsing[C]//Proc of the 2021 Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Stroudsburg, PA: ACL, 2021: 2760−2766

[3] Zelle J, Mooney R. Learning to parse database queries using inductive logic programming[C]//Proc of the 13th National Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 1996: 1050−1055

[4] Qin Bowen, Hui Binyuan, Wang Lihan, et al. A survey on text-to-SQL parsing: Concepts, methods, and future directions[J]. arXiv preprint, arXiv: 2208.13629, 2022

[5] Guo Jiaqi, Zhan Zecheng, Gao Yan, et al. Towards complex text-to-SQL in cross-domain database with intermediate representation [C]//Proc of the 57th Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2019: 4524−4535

[6] Wang Bailin, Shin R, Liu Xiaodong, et al. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers[C]//Proc of the 58th Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2020: 7567−7578

[7] Qi Jiexing, Tang Jingyao, He Ziwei, et al. RASAT: Integrating relational structures into pretrained Seq2Seq model for text-to-SQL[C]//Proc of the 2022 Conf on Empirical Methods in Natural Language Processing. Stroudsburg, PA: ACL, 2022: 3215−3229

[8] Cai Ruichu, Yuan Jinjie, Xu Boyan, et al. SADGA: Structure-aware dual graph aggregation network for text-to-SQL[J]. Advances in Neural Information Processing Systems, 2021, 34: 7664−7676

[9] Cao Ruisheng, Chen Lu, Chen Zhi, et al. LGESQL: Line graph enhanced text-to-SQL model with mixed local and non-local relations[C]//Proc of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th Int Joint Conf on Natural Language Processing. Stroudsburg, PA: ACL, 2021: 2541−2555

[10] Rajkumar N, Li R, Bahdanau D. Evaluating the text-to-SQL capabilities of large language models[J]. arXiv preprint, arXiv: 2204.00498, 2022

[11] Pourreza M, Rafiei D. DIN-SQL: Decomposed in-context learning of text-to-sql with self-correction[J]. Advances in Neural Information Processing Systems, 2023, 36: 36339−36348

[12] Gao Dawei, Wang Haibin, Li Yaliang, et al. Text-to-SQL empowered by large language models: A benchmark evaluation[J]. Proceedings of the VLDB Endowment, 2024, 17 (5): 1132−1145

[13] Wang Bing, Ren Changyu, Yang Jian, et al. MAC-SQL: A multi-agent collaborative framework for text-to-SQL[C]//Proc of the 31st Int

Conf on Computational Linguistics. Stroudsburg, PA: ACL, 2025: 540−557

[14] Dong Xuemei, Zhang Chao, Ge Yuhang, et al. C3: Zero-shot text-to-SQL with ChatGPT[J]. arXiv preprint, arXiv: 2307.07306, 2023

[15] Pourreza M, Rafiei D. DTS-SQL: Decomposed text-to-SQL with small large language models[C]//Findings of the Association for Computational Linguistics (EMNLP 2024). Stroudsburg, PA: ACL, 2024: 8212−8220

[16] Li Haoyang, Zhang Jing, Liu Hanbing, et al. CodeS: Towards building open-source language models for text-to-SQL[J]. Proceedings of the ACM on Management of Data, 2024, 2 (3): 1−28

[17] Talaei S, Pourreza M, Chang Yuchen, et al. Chess: Contextual harnessing for efficient SQL synthesis[J]. arXiv preprint, arXiv: 2405.16755, 2024

[18] Liu Ruiheng, Zhang Yu, Yang Bailong, et al. Robust and resource-efficient table-based fact verification through multi-aspect adversarial contrastive learning[J]. Information Processing & Management, 2024, 61 (6): 103853

[19] Liu Ruiheng, Zhang Jinyu, Song Yanqi, et al. Filling memory gaps: Enhancing continual semantic parsing via SQL syntax variance-guided LLMs without real data replay[C]//Proc of the 39th AAAI Conf on Artificial Intelligence and 37th Conf on Innovative Applications of Artificial Intelligence and 15th Symp on Educational Advances in Artificial Intelligence. Palo Alto, CA: AAAI, 2025: 24641−24649

[20] Liu Ruiheng, Zhang Jinyu, Song Yanqi, et al. Discarding the crutches: Adaptive parameter-efficient expert meta-learning for continual semantic parsing[C]//Proc of the 31st Int Conf on Computational Linguistics. Stroudsburg, PA: ACL, 2025: 3560−3578

[21] Wu Kun, Wang Lijie, Li Zhenghua, et al. Data augmentation with hierarchical SQL-to-question generation for cross-domain text-to-SQL parsing[C]//Proc of the 2021 Conf on Empirical Methods in Natural Language Processing. Stroudsburg, PA: ACL, 2021: 8974−8983

[22] Hu Yiqun, Zhao Yiyun, Jiang Jiarong, et al. Importance of synthesizing high-quality data for text-to-SQL parsing[C]//Findings of the Association for Computational Linguistics (ACL2023). Stroudsburg, PA: ACL, 2023: 1327−1343

[23] Yang Wei, Xu Peng, Cao Yanshuai. Hierarchical neural data synthesis for semantic parsing[J]. arXiv preprint, arXiv: 2112.02212, 2021

[24] Ye Jiacheng, Li Chengzu, Kong Lingpeng, et al. Generating data for symbolic language with large language models[C]//Proc of the 2023 Conf on Empirical Methods in Natural Language Processing. Stroudsburg, PA: ACL, 2023: 8418−8443

[25] Niu Yilin, Huang Fei, Liu Wei, et al. Bridging the gap between synthetic and natural questions via sentence decomposition for semantic parsing[J]. Transactions of the Association for Computational Linguistics, 2023, 11: 367−383

[26] Yu Tao, Wu C, Lin X, et al. Grappa: Grammar-augmented pre-training for table semantic parsing[J]. arXiv preprint, arXiv: 2009.13845, 2020

[27] Baumgartner D, Kornuta T. SynQL: Synthetic data generation for in-domain, low-resource text-to-SQL parsing[C]//Proc of the 3rd Table Representation Learning Workshop (NeurIPS). New York: Curran Associates Inc, 2024: 226−244

[28] Tai Changyu, Chen Ziru, Zhang Tianshu, et al. Exploring chain of thought style prompting for text-to-SQL[C]//Proc of the 2023 Conf on Empirical Methods in Natural Language Processing. Stroudsburg, PA: ACL, 2023: 5376−5393

[29] Zhang Hanchong, Cao Ruisheng, Chen Lu, et al. ACT-SQL: In-Context learning for text-to-SQL with automatically-generated chain-of-thought[C]//Findings of the Association for Computational Linguistics (EMNLP 2023). Stroudsburg, PA: ACL, 2023: 3501−3532

[30] Madaan A, Tandon N, Gupta P, et al. Self-refine: Iterative refinement with self-feedback[J]. Advances in Neural Information Processing Systems, 2023, 36: 46534−46594

[31] Lee C, Polozov O, Richardson M. KaggleDBQA: Realistic evaluation of text-to-SQL parsers[C]//Proc of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th Int Joint Conf on Natural Language Processing. Stroudsburg, PA: ACL, 2021: 2261−2273

[32] Zhang Yi, Deriu J, Katsogiannis-Meimarakis G, et al. Science-Benchmark: A complex real-world benchmark for evaluating natural language to SQL systems[J]. Proceedings of the VLDB Endowment, 2023, 17 (4): 685−698

[33] Yu Tao, Zhang Rui, Yang Kai, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task[C]//Proc of the 2018 Conf on Empirical Methods in Natural Language Processing. Stroudsburg, PA: ACL, 2018: 3911−3921

[34] Touvron H, Lavril T, Izacard G, et al. Llama: Open and efficient foundation language models[J]. arXiv preprint, arXiv: 2302.13971, 2023

[35] Ouyang Long, Wu J, Jiang Xu, et al. Training language models to follow instructions with human feedback[C]//Proc of the 36th Int Conf on Neural Information Processing Systems. New York: Curran Associates Inc, 2022: 27730−27744

[36] Chen M, Tworek J, Jun H, et al. Evaluating large language models trained on code[J]. arXiv preprint, arXiv: 2107.03374, 2021

[37] Guo Daya, Zhu Qihao, Yang Dejian, et al. DeepSeek-Coder: When the large language model meets programming —The rise of code intelligence[J]. arXiv preprint, arXiv: 2401.14196, 2024

[38] Team Qwen. Qwen2 technical report[J]. arXiv preprint, arXiv: 2407.10671, 2024

[39] Roziere B, Gehring J, Gloeckle F, et al. Code Llama: Open foundation models for code[J]. arXiv preprint, arXiv: 2308.12950, 2023

[40] Guo Daya, Yang Dejian, Zhang Haowei, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning[J]. arXiv preprint, arXiv: 2501.12948, 2025

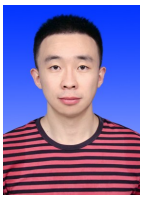
[41] Li Haoyang, Wu Shang, Zhang Xiaokang, et al. OmniSQL: Synthesizing high-quality text-to-SQL data at scale[J]. Proceedings of the VLDB Endowment, 2025, 18 (11): 4695−4709

[42] Wang Dingzirui, Dou Longxu, Zhang Xuanliang, et al. DAC: Decomposed automation correction for text-to-SQL[C]//Findings of the Association for Computational Linguistics (EMNLP 2025). Stroudsburg, PA: ACL, 2025: 385−402



Ren Qianhao, born in 2003. Master candidate. His main research interests include natural language processing and natural language interface to databases.

任千昊, 2003 年生。硕士研究生。主要研究方向为自然语言处理、数据库自然语言接口。



Liu Ruiheng, born in 1997. PhD. His main research interests include natural language processing and semantic parsing.

刘睿珩, 1997 年生。博士。主要研究方向为自然语言处理、语义解析。



Zhang Yu, born in 1972. PhD, professor. His main research interests include natural language processing, question answering systems, and information retrieval.

张　宇, 1972 年生。博士，教授。主要研究方向为自然语言处理、问答系统、信息检索。