中国科学技术大学

硕士学位论文

基于大语言模型的 Text-to-SQL 方法研究与应用

作者姓名： 申一凯

学科专业： 计算机科学与技术

导　　师： 刘淇 教授

完成时间： 二〇二五年四月十一日

University of Science and Technology of China

A dissertation for master's degree

Research and Application of Text-to-SQL Methods with Large Language Models

Author: Yikai Shen

Speciality: Computer Science and Technology

Supervisor: Prof. Qi Liu

Completion date: April 11, 2025

摘要

随着信息的爆炸式增长，关系型数据库的应用场景和存储数据规模都在不断增加。Text-to-SQL 任务旨在实现从自然语言问题到结构化查询语言（SQL）的自动化转换。它在自然语言序列与结构化表格数据、非专业用户与数据库系统之间搭建起交互的桥梁，极大地提高了数据处理的效率，为智能数据库服务、自动数据分析及数据库问答等众多领域的广泛应用提供了有力支持。

近年来，大语言模型在 Text-to-SQL 任务上取得了显著进展，大幅提高了将自然语言查询自动转化为语义等价的 SQL 的能力。不过，在实际应用过程中，基于大语言模型的 Text-to-SQL 任务仍面临着一系列不容忽视的问题：首先，大语言模型在处理复杂查询时经常出现错误，生成的 SQL 即使语法正确，也可能会与预期含义不符。通过多智能体协作分解任务，有效纠正模型输出，成为了当前研究的关键部分；其次，将大语言模型的推理能力泛化到轻量化模型，使其能准确完成端到端的输出，是本任务在现实场景中部署应用的难点；最后，在与数据库应用高度相关的表格问答任务，尤其是大型自由格式表格问答中，大语言模型回答的准确率仍存在较大提升空间。借助 SQL 语句进行检索计算辅助表格推理，对智能数据库系统的构建存在重大意义。

针对上述挑战，本文从以下三个层面开展了关于 Text-to-SQL 任务的研究：

（1）从提示工程的角度出发，本文提出了一种基于多智能体协作和模式对齐的 Text-to-SQL 算法，以解决大语言模型在复杂 Text-to-SQL 任务中面临的多种因素制约，包括整体框架的不完整、推理的性能损失以及模型幻觉等问题。它将任务划分为三个主要模块：预处理、推理和后处理，并在模块之间设置对齐机制。该架构通过对智能体的输入和输出进行对齐，减少了指令遵循失败和幻觉的出现。此外，本文还设计了一种中间语言，对传统的结构化思维链（CoT）进行优化以实现更准确的复杂推理。这些方法显著提高了大语言模型在 Text-to-SQL 任务中的性能。

（2）从微调的角度出发，本文提出了一种基于数据增强和多任务协作的 Text-to-SQL 算法，使用与 SQL 生成相关的多种合成训练数据进行多任务监督微调，以提升轻量化模型在任务中的性能。与现有的直接微调方法不同，本文引入了额外的模式链接和错误纠正任务，增强模型对 SQL 语法的理解，提高其生成高质量 SQL 查询的能力。错误纠正任务从提取的的错误样本和纠错方法中学习，可以有效识别和纠正多类句法和语义错误。本文在轻量化模型上通过多个 SQL 相关任务的协作实现了可与大型模型相较的性能表现。

（3）本文提出了一种基于 SQL 分解的表格问答算法，以增强大语言模型处

理大型自由格式表格的能力。它先根据表格的关键信息生成 SQL 查询，并以此指导表格分解，去除噪声，在子表格上更好地生成答案。此外，本文还引入动态的验证器优化分解路径，以提高分解的准确性。SQL 和表格操作的转换，增强了检索与定位关键信息的能力，同时兼顾了文本推理的灵活性，在处理大型自由格式表格时性能显著。

总之，针对 Text-to-SQL 任务，本文从大语言模型的两条技术路线——提示工程和微调出发，分别给出了可行的解决方案，并展示了在高度相关的表格问答任务上应用 Text-to-SQL 提升整体性能的范例，为推动智能数据库系统的发展做出了自己的贡献。

关键词：Text-to-SQL 解析；表格问答；大语言模型；提示工程，有监督微调

ABSTRACT

With the explosive growth of information, the application scenarios of relational databases and the scale of stored data are constantly increasing. Text-to-SQL task aims to achieve the automated conversion from natural language questions to Structured Query Language(SQL). It builds an interactive bridge between natural language sequences and structured tabular data, as well as between non-professional users and database systems. This greatly improves the efficiency of data processing and provides strong support for a wide range of applications in many fields, such as intelligent database services, automated data analysis, and database question answering.

In recent years, Large Language Models(LLMs) have made remarkable progress in the field of Text-to-SQL, significantly improving the ability to automatically convert natural language queries into semantically equivalent SQL. However, in the actual application process, Text-to-SQL task based on LLMs still faces a series of problems that cannot be ignored: Firstly, LLMs often have hallucinations when dealing with complex queries. Even if the generated SQL is grammatically correct, it may not match the intended meaning. Decomposing tasks through multi-agent collaboration and effectively correct the model output has become a key part of current research. Secondly, generalizing the reasoning ability of LLMs to lightweight models and enabling them to accurately perform end-to-end output, is a difficulty in the practical application of the task in real-world scenarios. Finally, in the table question answering task that is highly relevant to database applications, especially in large free-form tables, there is still much room for improvement in the accuracy of the answers provided by LLMs. Using SQL statements for retrieval and calculation to assist in table reasoning is of great significance for the construction of intelligent database systems.

In response to the above challenges, three key contributions are presented:

(1) From the perspective of prompt engineering, the study proposes a Text-to-SQL algorithm based on multi-agent collaboration and schema alignment to address the constraints of LLMs in complex Text-to-SQL tasks. These constraints include incomplete overall frameworks, reasoning performance degradation, and model hallucinations. The algorithm divides the task into three main modules: preprocessing, reasoning, and post-processing, with alignment mechanisms set between modules. By aligning the inputs and outputs of agents, this architecture reduces instances of instruction-following failures and hallucinations. Additionally, the study designs an intermediate language to

optimize traditional structured thought chains for more accurate complex reasoning. These methods significantly enhance the performance of LLMs in Text-to-SQL task.

(2) From the perspective of fine-tuning, the study proposes a Text-to-SQL algorithm based on data augmentation and multi-task collaboration. It uses various synthetic training data related to SQL generation for multi-task supervised fine-tuning, aiming to enhance the performance of lightweight models in the task. Different from existing direct fine-tuning methods, the study introduces additional schema linking and error correction tasks to strengthen the model's understanding of SQL syntax and improve its ability to generate high-quality SQL queries. The error correction task learns from extracted error samples and correction methods, which can effectively identify and correct multiple types of syntactic and semantic errors. Through the collaboration of multiple SQL-related tasks on lightweight models, the study achieves performance comparable to larger models.

(3) The study proposes a table question-answering algorithm based on SQL decomposition to enhance the ability of LLMs to process large free-form tables. It first generates SQL queries based on the key information of the table, which guide table decomposition to remove noise and facilitate better answer generation on sub-tables. Additionally, the study introduces a dynamic validator to optimize the decomposition path and improve decomposition accuracy. The transformation between SQL and table operations enhances the capability to retrieve and locate key information while maintaining the flexibility of text reasoning, achieving significant performance improvements in handling large free-form tables.

In conclusion, for the Text-to-SQL task, this study starts from two technical routes of LLMs—prompt engineering and fine-tuning, and provides feasible solutions respectively. It also demonstrates an example of applying Text-to-SQL to improve the overall performance in the highly relevant table question answering task, making its own contribution to the development of intelligent database systems.

KEY WORDS: Text-to-SQL Parsing, Table Question Answering, Large Language Models, Prompt Engineering, Supervised Fine-Tuning

目 录

插图清单

附表清单

第 1 章　绪论

1.1　研究背景和意义

随着信息技术的高速发展，人们越来越依赖关系数据库来管理和分析大量结构化信息。这些数据库是许多现代系统的关键部分，从商业智能到客户关系管理。随着数据量的增加，查询、提取和理解此信息的需求也随之增加[1]。让缺乏数据库专业知识的普通用户可以使用自然语言与结构化数据进行交互，是如今构建数智化系统的重要指标，也是人机协同决策体系的核心基础[2]。在数字化转型浪潮中（如商业分析、医疗信息化、金融风控等领域），将非结构化自然语言查询精准转化为可执行的结构化查询语句（SQL），已成为释放数据价值、提升决策效率的关键技术环节[3]。

在自然语言处理与数据库系统的交叉研究领域，开发能够自动将自然语言问题转换为标准 SQL 的智能系统，是一个持续受到关注且极具挑战性的前沿方向。这一目标的实现要求研究者不断突破语义理解、模式关联、逻辑表达等多重技术瓶颈，提升模型对复杂查询意图的解析能力、对数据库模式结构的认知能力，以及对多步推理过程的建模能力。随着企业级数据规模的指数级增长，构建高效可靠的 Text-to-SQL 系统不仅是技术演进的必然需求，更是实现数据民主化战略的重要技术路径。

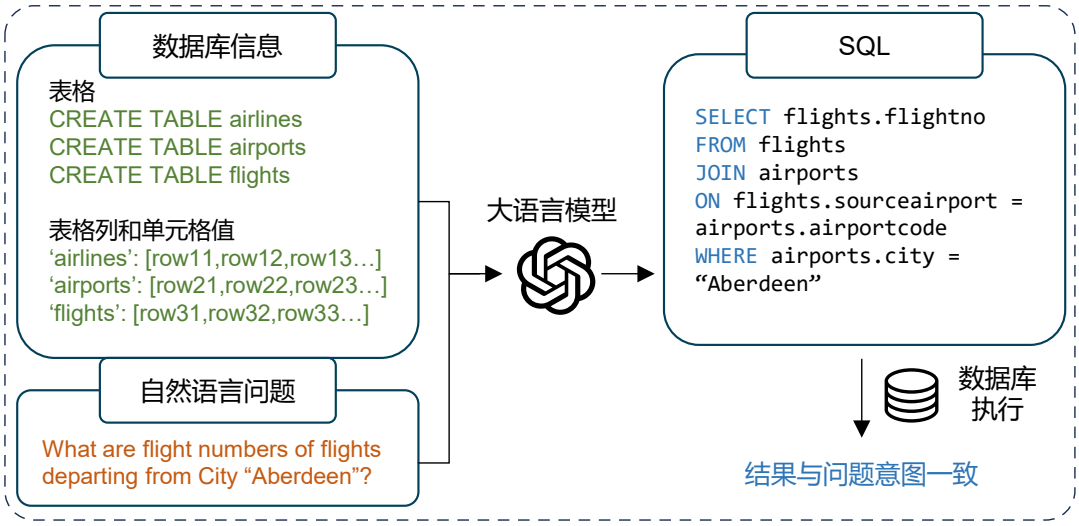

图 1.1　Text-to-SQL 任务示例

尽管大语言模型在语义理解与推理问答任务中展现出显著优势[4]，Text-to-SQL 任务对现有模型而言仍存在严峻挑战[5]。任务的一个示例如图 1.1所示，大语言模型要同时处理文本形式的问题和数据库中各个结构化表格的关键信息，并将其解析成一个语法正确，且查询逻辑与用户意图精确匹配的 SQL 查询语句。相

较于常规文本，涉及到结构化数据的查询往往包含隐式的领域知识、动态变化的表达方式，以及嵌套在口语化描述中的复杂查询逻辑。如何有效捕捉查询文本与数据库模式间的深层语义关联，建立鲁棒的跨模态表征映射，仍是亟待突破的技术难点。

先前基于大语言模型的解决方案虽然在基准数据集上取得了可观进展，但主要聚焦于单表简单查询场景，在应对多表连接、嵌套子查询、复杂条件组合等实际需求时，仍可能存在模式链接错误、语义消歧失效、逻辑结构混乱等问题。即使经过专门调优，在处理跨领域复杂查询时也常面临性能显著下降的困境[6]。

在众多应用场景中，跨表复杂查询[7-9]因其模式设计相对复杂、查询需求更为多样性、数据关联更为紧密，成为检验 Text-to-SQL 系统实用价值，以及大语言模型推理能力的关键问题。本研究以基于大语言模型的 Text-to-SQL 为主题，通过提示工程和监督微调这两种技术路线，研究不同场景下大语言模型应用于任务的解决方案。同时，本文还探讨了与数据库智能化系统密切相关的表格问答任务中 Text-to-SQL 的应用方式。值得强调的是，Text-to-SQL 技术作为数据库智能化系统自然语言接口的核心组件，正在为企业带来全新的数据交互范式。在此背景下，提升整个系统的可用性不仅具有学术创新价值，更对推动产业智能化转型具有战略意义。接下来，1.2 节将介绍相关领域的研究现状及面临的主要挑战，1.3 节介绍本文的主要工作和贡献，最后在 1.4 节中介绍本文组织结构。

1.2　研究现状和研究挑战

1.2.1　基于提示工程的 Text-to-SQL 算法

提示工程的研究源自于大语言模型强大的指令跟随与上下文学习能力，无需微调，仅需一段指令文字说明和可选的样本示例，大语言模型即可按照要求完成指示的任务。在基于提示工程的设计中，C3[10]开发了一种系统的零样本 Text-to-SQL 方法，其中包括模式链接、清晰提示等几个关键组件。DIN-SQL[11]通过区分问题难度并相应地调整特定提示来解决 Text-to-SQL 任务。DAIL-SQL[12]通过检索结构相似的问题或 SQL 作为小样本来协助模型生成 SQL，而 MAC-SQL[13]则通过任务分解和重组来改进功能。MCS-SQL[14]通过使用多组投票提示来增强 Text-to-SQL 任务的稳定性。

面对一个专业领域的复杂问题，往往需要涉及到隐式领域知识、复杂查询逻辑，以及 SQL 语法结构正确性的多方面考量。然而，现有的研究大多只针对整个流程中的局部进行优化，即使将任务拆分成多个智能体协作，也缺乏对智能体连贯性和耦合性的保障。同时，在最关键的推理阶段，现有的方法并不能很好地分解复杂问题以降低生成难度，虽然有基于思维链[15]和问题分解[11]的研究，但

它们受人类的先验知识影响，难以发挥大语言模型真正的推理能力。除此之外，大语言模型存在着固有的幻觉。在多智能体协作的流水线中，这种幻觉所导致的偏差可能会传播累积，进而影响最终结果。制订统一的 SQL 生成框架，降低复杂问题生成相应 SQL 的难度，提高大模型遵循指令的能力，已成为基于提示工程的算法设计中共同面临的挑战。

1.2.2　基于微调的 Text-to-SQL 算法

早在大语言模型出现之前，监督微调的研究便已经成为 Text-to-SQL 任务中的热点。研究者们将 Text-to-SQL 视为序列到序列任务的一个典范，依靠标注的大量 {问题，SQL} 对在预训练语言模型中微调实验结果。RESDSQL[16]将 SQL 中的模式与骨架解耦，提出先生成骨架再生成完整 SQL 查询的任务流程。Graphix[17]将 SQL 转换为图，把节点间的关系编码纳入微调参数中。

大语言模型出现之后，通过预训练或监督微调将通用大语言模型转化为专用大语言模型成为主要的研究方向[18-21]。然而，大多数基于微调的方法仅在监督微调阶段纳入训练集中给定的 SQL 生成任务，这使得参数量级较小的模型能力高度依赖训练集，难以在广泛知识边界中具有泛化性。此外，在单一 SQL 生成任务上训练大语言模型，在理解指令方面存在性能下降的巨大风险，这可能会降低模型在 SQL 生成之外的其他重要 SQL 相关任务中的有效性。考虑到现有训练方法的局限性，如何以轻量化模型为基础训练 Text-to-SQL 专用模型，使其在多个领域中仍拥有泛化性和相对较优的性能表现，是关乎到将来智能数据库系统能否在日常生活中部署和应用的难点。

1.2.3　通用表格问答算法

常见的表格理解任务包括表格问答[22-23]、表格事实验证[24-25]、表格到文本生成[26-27]。在这些任务中，表格问答是最具挑战性的任务之一，它要求模型同时理解自然语言问题和相应的自由格式表格数据。随着大语言模型展现出强大的性能，大量研究致力于自由格式表格问答[28]。然而大多数现有研究面向的表格规模较小，当这些研究用于大型自由表格时，往往会遇到输入开销大、噪声敏感、推理幻觉严重等一系列挑战，需要程序辅助推理（如 SQL、Python）作为中间工具，减少输入开销，去除大规模数据中的噪声。同时，相较于数据库中的结构化表格，自由格式表格没有预定义的模式限制，范围更广，这使得基于结构化查询生成 SQL 的方法[22,29]无法直接应用。如何调整 SQL 的使用策略，使其有助于整个表格问答任务，是打破数据库中的结构化表格和日常生活的自由格式表格的边界，将智能化数据库系统适用于更广阔领域的关键要素。

1.3　研究内容和主要贡献

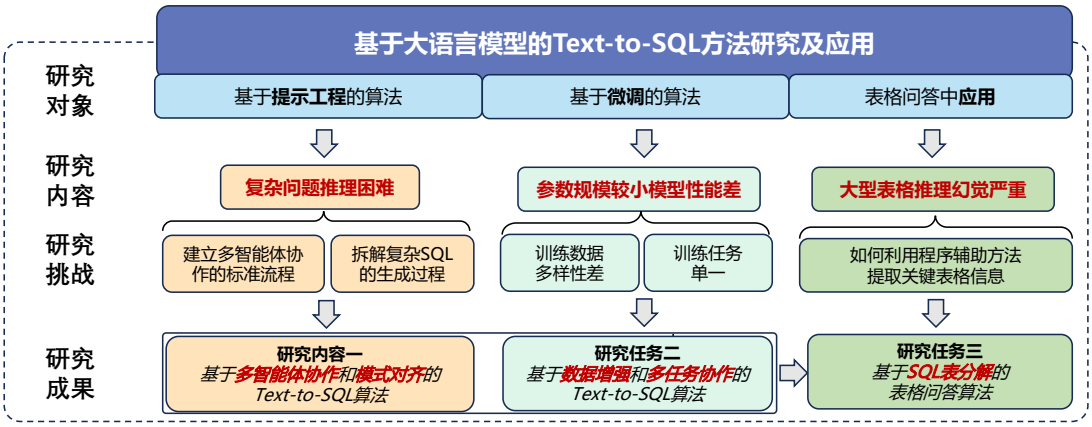

图 1.2　研究整体框架

本文的研究框架如图 1.2所示，简要来说，为了实现基于大语言模型的 Text-to-SQL 方法，本文首先从提示工程的技术路线出发，设计了一套多智能体协作与模式对齐的算法，探索任务性能的上界。然后，这种协作模式启发了微调的新范式，引出了基于数据增强与多任务协作的微调算法，研究任务在算力受限场景中的部署方案。最后，本文将研究领域由数据库拓展到日常生活中的自由格式表格，在表格问答任务中应用 Text-to-SQL，提出了通用的表格问答算法，它可以基于 SQL 进行表分解，在降低输入开销的时候提高整体性能，展现了 Text-to-SQL 在广泛场景的应用价值。

- 基于提示工程的 Text-to-SQL 算法：面对任务中的复杂推理问题，目前主流的提示工程方法没有形成相对统一的框架，大多只针对整个流程中的局部进行优化，即使将任务分成多个智能体协作，也缺乏对智能体连贯性和耦合性的保障。为此，本文提出了多智能体协作与模式对齐算法 SA-SQL，定义了预处理、推理、后处理三个模块并设计相应智能体，构建了 Text-to-SQL 的统一框架。在模块之间引入对齐策略，以缓解中间参数传递的传递损失及最终输出结果的幻觉现象。同时，本文将中间表示引入推理智能体的思维链，提高复杂 SQL 生成的准确率。在多个智能体的协同作用下，SA-SQL 在多个数据集上展现出优秀的推理性能。
- 基于微调的 Text-to-SQL 算法：基于提示工程的方法虽然在参数规模较大、推理能力较强的大语言模型上表现良好，但应用于轻量化大语言模型时有效性会降低。同时现有的基于微调的方法仅在监督微调阶段纳入 SQL 生成任务，存在理解指令性能下降的风险。考虑到上述方案的局限性，本文提出了数据增强和多任务协作的 Text-to-SQL 算法 EnhancedSQL。具体来说，本文将与 Text-to-SQL 密切相关的两个任务——模式链接和错误校正加入训练过程，增强模型对 SQL 语法的理解。各项任务的训练数据依据不同的

策略合成，以提升模型在不同领域中的泛化能力和针对常见错误的纠错能力。在推理过程中采用多任务协作提示的方法，选择性地逐步生成 SQL，降低复杂 SQL 生成中出现幻觉的风险。EnhancedSQL 在轻量化模型中取得了优异的性能表现，为日常生活中 Text-to-SQL 解析器的部署落地提供了可行方案。

- 通用表格问答算法：表格问答任务可以通过将表格全部信息输入大语言模型来生成相应答案，然而在大型表格问答场景中，大语言模型受输入令牌长度限制和噪声干扰表现性能不佳，幻觉现象严重。如何有效利用程序辅助方法过滤噪声，提取关键表格信息是一项重大挑战。此外，现有的仅靠 Text-to-SQL 的方法不足以解决自由格式表格问答问题。为此，本文提出了基于 SQL 进行表分解的表格问答算法 DesTab，利用生成的 SQL 查询映射表格分解操作，提取无噪声的关键性子表格，显著提高了答案生成的准确性，在大型表格问答数据集上相较于传统的全表输入方法性能更优且极大减少了输入开销。

1.4　本文组织结构

本文由六个章节组成，其组织结构如下：

第一章为绪论部分。开篇介绍了基于大语言模型的 Text-to-SQL 研究背景与意义，梳理了该背景下国内外的研究现状。随后，指出了现有研究工作存在的不足，概括本文的研究动机与贡献。

第二章详细介绍了使用大语言模型完成 Text-to-SQL 任务的相关工作。首先，从大语言模型的两条技术路线——提示工程与微调出发阐述了当前工作的主要类别。随后，对各类方法上重点关注的要素进行了分类枚举，同时还介绍了与 Text-to-SQL 紧密相关的表格问答任务的发展历程，以及现有的借助 SQL 等工具实现表格分解的方法。

第三章针对当前工作在复杂问题上推理困难的问题，提出了一种基于提示工程的多智能体协作和模式对齐算法 SA-SQL。它通过建立多智能体协作的标准流程，在各个智能体中引入关键策略，缓解了现有工作在协作中的幻觉与错误累积。同时它提出了一种拆解复杂 SQL 生成的思维链策略，对复杂问题的推理提供了可靠的解决方案，并通过多项基准测试证明了它的有效性。

第四章针对当前轻量化模型性能表现差的问题，提出一种基于微调的数据增强与多任务协作算法 EnhancedSQL。它利用数据增强强化了模型在不同领域中的泛化性和典型错误中的纠错能力，同时引入额外训练任务和多任务协作提示，使轻量化模型展现出和大型模型相当的性能，为算力受限的实际场景部署提

供了参考。

第五章针对大型表格问答任务中幻觉严重的问题，提出一种基于 SQL 表分解的大型表格问答算法 DesTab。它将生成的 SQL 作为表格分解的中间件，将 SQL 映射为多种表格操作策略，极大降低了表格问答任务中的输入开销。基于 SQL 的方法在表格问答任务中的成功打破了数据库结构化表格与自由格式表格的边界，统一了表格与数据库的任务场景。

第六章对本文作总结，对未来可能的发展方向作展望。

第 2 章　相关工作和基础知识

本章主要介绍 Text-to-SQL 任务的相关工作以及基础知识。2.1 节介绍 Text-to-SQL 任务的发展历程，并引出基于大语言模型的任务现有的两条技术路线——提示工程和微调[30]。在 2.2、2.3 节分别对两种技术路线中主要的研究热点以及相关研究工作进行了总结，在 2.4 节中对与 Text-to-SQL 关系密切的表格问答任务进行了简要介绍。通过以上叙述内容，本章阐述了各项任务的发展现状，为接下来 3-5 章介绍本文各项工作打下基础。

2.1　Text-to-SQL 任务发展历程

Text-to-SQL 的目的是将自然语言问题自动转化为相应的 SQL 查询语句。它弥合了非专家用户和数据库系统之间的差距，极大地提高了数据处理的效率，并有助于更广泛的应用程序，例如智能数据库服务，自动数据分析和数据库问答。数据库和自然语言处理社区对 Text-to-SQL 进行了广泛的研究。在大语言模型普及之前，Text-to-SQL 方法主要有两大类别。一类方法采用序列到序列（Seq2Seq）模型[31]，在该模型中，编码器用于捕捉自然语言问题及相应表结构的语义，解码器则基于编码后的问题表征逐个词元地生成 SQL 查询语句。在这类方法中，IRNet[32]提出了介于自然语言问题和 SQL 表达式的中间语言 SemQL，使用基于语法的神经网络模型去合成 SemQL。Seq2SQL[29]利用分类任务对分解后的架构进行槽位内容的预测和填写。RESDSQL[16]则通过解耦模式链接和骨架生成，促使解码器生成符合 SQL 语法结构的表述。另一类方法是对 BERT[4]等预训练语言模型进行微调，这些模型利用大规模文本集合中蕴含的丰富知识，可以有效提升下游 Text-to-SQL 解析任务的性能。此外，为了缩小 Text-to-SQL 研究及其现实世界部署之间的差距，研究者们发布了若干大规模的基准数据集，包括 WikiSQL[29]、Spider[8]、BIRD[33]等。数据集中的样例从单表问答演化为多表问答，再到不同专业领域的复杂问题，对 Text-to-SQL 方法的性能要求也不断增加。

大语言模型自问世以来，已经成为自然语言处理和机器学习领域的一个里程碑。在过去的几年中，研究者们基于 Transformer 架构不断扩大预训练语言模型的参数规模以及训练数据量[34]。随着参数量的指数级增长，研究人员发现大语言模型具备了一些令人惊讶的能力，即所谓的“涌现能力”[35]，而这些能力在参数较少、架构较为简单的预训练语言模型中是不存在的。涌现能力的一个例子是少样本学习[36]，表现为大语言模型能够在提示中通过几个合适的任务示例来完成下游任务，而无需进一步训练。另一个例子是指令跟随能力[36]，借助这一

能力，大语言模型已被证明能够对描述未知任务的指令做出恰当的回应。

由于大语言模型所具有的涌现能力[35]，以及大语言模型基于输入提示逐步生成具有最高概率的下一个词的基本运行原理[34]，提示工程成为将大语言模型应用于下游任务的两大主要方向之一。提示工程的代表性方法包括检索增强生成（RAG）[37]、少样本学习[36] 以及推理[38-40]。将大语言模型应用于下游任务的另一个方向被称为微调，它遵循预训练语言模型的“预训练-微调”学习范式[34]，旨在提升特定领域的性能并解决隐私问题。一般的微调过程主要包括数据准备、模型选择、模型微调和模型评估。

Text-to-SQL 是自然语言处理和数据库领域中一项具有挑战性的任务，它涉及将给定关系数据库上的自然语言问题映射为 SQL 查询语句。大语言模型的出现也给这项任务带来了变革。按照研究[30]中描述，本文将基于大语言模型的 Text-to-SQL 所使用的方法分为两类：提示工程和微调。在基于提示工程的方法中，研究者们通常会设计一个结构良好的提示，其中包含各种组成部分，例如任务描述、数据库模式、问题以及额外的知识，同时还会使用上下文学习和不同推理策略。在基于微调的方法中，研究者们通常会生成或收集 Text-to-SQL 的数据集，选择合适的预训练大语言模型以及诸如低秩适应（LoRA）[41]之类的微调方法，并比较微调前后的测试结果，以了解模型性能的变化情况。传统的 Text-to-SQL 方法与基于大语言模型的方法主要有两个方面的区别：（1）传统的 Text-to-SQL 方法需要进行训练，而大语言模型往往可以绕开这一要求。借助大语言模型的指令跟随能力[42]，大语言模型能够在合适的指令和信息的辅助下完成 Text-to-SQL 的任务。（2）传统方法中的编码器和解码器可以采用多种不同的架构来设计，比如长短期记忆网络（LSTM）[43]、Transformer 架构[44]，甚至图神经网络（GNN）[45]。相比之下，大语言模型采用统一的基于 Transformer 的架构，这不仅使模型更容易进行规模扩展，而且也让实现过程更加简洁流畅。

2.2　基于提示工程的 Text-to-SQL

提示工程，也被称为上下文学习，指的是构建大语言模型在某种程度上能够理解的指令代替训练过程。从研究者的角度来看，这意味着在与大语言模型交互时，通过设计提示词来定制大语言模型在某些特定任务上的输出。由于 Transformer 架构中自回归解码的特性[46]，大多数大语言模型会基于当前所有可见的前文文本（即上下文）来预测后续文本。本文将自回归解码描述为公式 2.1，其中 y_t 表示大语言模型即将输出的下一个标记，而 $\mathbf{x}$ 表示用户给出的提示标记。作为上下文的提示词的设计会影响所有即将生成的标记的概率分布，进而影响最

终的生成结果。

$$y_t = \arg\max P(y_t|y_{1:t-1}, \mathbf{x}) \tag{2.1}$$

在对相关工作进行全面研究后，根据作用阶段的不同，本文将 Text-to-SQL 任务中的提示工程方法分为以下三个模块：

- **预处理**：在实际场景中，从事 Text-to-SQL 任务的专家常常会因问题描述的不清晰以及数据库模式的模糊性而感到困扰。这种情况同样适用于大语言模型，因此需要对 Text-to-SQL 的问题进行预处理。本文将提示工程方法的预处理部分将包括三个部分：首先，对问题描述进行文本序列化表示；其次，对数据库模式进行选择性关联；最后，引入额外的 SQL 知识或外部知识。
- **推理**：推理是指在给定用户问题和相应数据库模式的情况下，生成对应的 SQL 查询语句。从逻辑上讲，这个阶段可以分为两个部分：推理流程设计和示例选择。从预处理过渡到 SQL 生成，大多数研究工作会以自定义的方式或基于推理模式来设计其推理过程，并决定是否使用示例。本文将首先介绍 Text-to-SQL 任务中的流程设计，包括一些常用的推理策略，如思维链（Chain-of-Thought）[38]、从少到多（Least-to-Most）[39] 以及分解（Decomposition）。在此之后，本文将介绍示例方法，包括零样本方法和少样本方法。在少样本方法部分，本文将强调示例风格和示例选择的重要性。
- **后处理**：为了提高基于大语言模型的 Text-to-SQL 方法的性能和稳定性，在推理之后对生成的结果进行进一步优化是一种可行的选择。本文将这些操作称为后处理。这一任务中常见的后处理方法包括自我校正（Self-Correction）[47] 以及一致性方法[48]（Self-Consistency）和交叉一致性[49-50]（Cross-Consistency）。

2.2.1　预处理

在 Text-to-SQL 任务的起始阶段，有必要在提示词中清晰且全面地描述解决问题所需的所有信息。这些信息主要包括问题的表示格式、数据库模式以及一些与任务相关的知识。在预处理部分，本文将对上述每一个信息部分进行介绍。

1. 问题表示

Text-to-SQL 任务中的“问题”包含两个部分：自然语言的问题陈述以及相关数据库的必要信息。两种常见的组织格式是 OpenAI 模板和“创建表”模板，如图 2.1所示。前者出现在 OpenAI 的官方演示中，可以简洁地概括问题涉及的数据库信息；而后者是创建数据库使用的标准程序语言，相较于前者包含的说明信息更多，且和预训练过程中代码数据更为贴切。

除了问题和表格关键信息的布局，来自真实数据库内容的样本数据对于问

```
### SQLite SQL tables, with their
    properties
#
# schools(SchID, city)
# teachers(TechID, SchID)
# students(StuID, SchID, TechID)
# course(CourseID, SchID)
# grade(StuID, CourseID, score)
#
#
### {Question}
SELECT
```

(a) Openai Template

```
### SQLite SQL tables, with their
    properties
CREATE TABLE schools(
    SchID int primay key,
    city text
);
CREATE TABLE teachers(
    TechID int primay key,
    SchID int references schools(TechID)
);
### {Question}
SELECT
```

(b) Create Table

图 2.1　常见提示模板类型

题表示也很有帮助。研究[51]将提示表示为 `SELECT * FROM Table LIMIT X` 语句，选择表格前 X 行的值作为示例输入，X 根据经验通常取值为 3。一些研究工作[52-57] 甚至直接将样本数据放入提示词中。这样做的目的是让大语言模型理解数据库中的样本数据，并在生成 SQL 语句时符合数据本身的格式。研究[58]修改了样本数据的形式，将其按列列出数据，以明确枚举分类值。

2. **模式链接**

在现实生活的数据库应用常常会出现这样的情况：一个问题涉及多个表中的多列数据，但每个表仅使用特定的列。然而，冗余且不相关的模式项可能会分散大语言模型识别正确项的注意力，因此需要进行模式链接。模式链接是 Text-to-SQL 过程中的一个关键子任务，用于指定数据库中与给定查询短语相对应的表和列。在基于大语言模型的 Text-to-SQL 流程中，这一步骤至关重要。首先，它能缩短输入序列的长度。一方面，对于大型数据库而言，将所有表的信息都输入到大语言模型中是不切实际的；另一方面，通过缩短表模式的长度有可能提高大语言模型的关注度和有效性。其次，实验结果表明，基于大语言模型的 Text-to-SQL 任务中，相当一部分失败样例源于无法正确识别问题中提到的列名、表名或实体。通过使用模式链接技术可以提升任务上的整体性能，甚至有助于跨域泛化以及复杂查询的合成。除此之外，部分 SQL 涉及到数据库中值的聚合计算，或者文本的匹配查找，此时不仅需要输入表和列名，还要根据相似度高低选择合适的单元格值作为输入。如字符串的大小、数字的表示格式等都可能会对 SQL 在数据库中的准确执行有一定影响。

常见的模式链接技术可分为两类，即传统的模式链接方法和基于大语言模型的模式链接方法。前者通常使用 BERT[4] 等预训练模型，将模式链接设定为为分类任务进行训练。RESDSQL[16] 中根据数据集的模式信息训练了额外的交叉编码器，可以计算各列名与自然语言问题的相似度进行选择；而后者利用大语言模型的先验知识和推理能力，直接设计模式链接任务的范式。随着涉及到的专业领域越来越多，部分领域中需要对简练的表和列名表达添加自然语言说明，以完成

数据库范式和自然语言问题中实体的映射关系。

3. **知识注入**

除了问题表示和模式关联方法之外，为具有通用能力的大语言模型提供相关知识也是有益的。添加到提示中的知识可以看作是对当前任务描述的一种校准，为后续的 SQL 生成扫清障碍。本文将知识分为两类，一类是与 SQL 相关的知识，另一类是与问题本身及数据相关的知识。

- **SQL 知识**：与 SQL 相关的知识主要包括 SQL 关键字、SQL 语法以及常见的 SQL 编写习惯。向通用大语言模型的提示中添加与 SQL 相关的知识，可以对任务输入的风格进行指导校正，避免语义错误。C3[10] 专门校准了模型在 SQL 风格上的偏差，并在提示中添加了以下指令：仅在特定情况下使用 `COUNT (*)`，避免使用 `LEFT JOIN`、`IN` 和 `OR`，而是使用 `JOIN` 和 `INTERSECT`，并建议使用 `DISTINCT` 和 `LIMIT` 关键字。DIN-SQL[11] 和 DEA-SQL[56] 注意到，像 `JOIN`、`INTERSECT` 和 `IN` 等一些关键字表明了 SQL 查询的难度，因此他们根据自己的判断，针对当前问题的难度级别设计了不同的规范和提示。同样，Meta-SQL[59] 设计了三种查询元数据，以便能够充分表达查询的高级语义。元数据的类型包括操作符标签、难度值和正确性指示符。为了进一步优化评判模型的决策能力，SQLfuse[60] 补充了列举常见错误的校准提示。这些提示有助于预先解决潜在的错误步骤，提升模型生成更优 SQL 查询的能力。
- **外部知识**：还有一些来自外部环境的各类知识可能对 Text-to-SQL 任务有所帮助，因为一些行业术语或特定领域的词汇如果没有解释的话很难理解。Open-SQL[57] 使用 BIRD 数据集[33] 为每个查询提供的额外描述性信息，这些信息充当了人类理解和数据库结构之间的桥梁。CHESS[61] 使用上下文检索方法来提取数据库目录、表和列的描述及缩写，以获得更好的性能。SQLfuse[60] 专门设计了一个 SQL 评判器模块来确定最优的候选 SQL 查询，该模块从 GitHub① 上一系列复杂的 SQL 语句和模式中构建了一个外部 SQL 知识库，以便更好地提取外部环境中的知识。

2.2.2　推理

在给定以特定形式呈现的问题和模式后，接下来的步骤是生成该问题的潜在答案。考虑到 Text-to-SQL 任务的复杂性以及对准确性的高要求，让大语言模型直接针对问题生成 SQL 回复很难得到令人满意的答案。本文对相关工作方法进行研究总结，发现主要有两种技术有助于生成正确且高质量的 SQL 语句，即推理过程和示例选择。

①https://github.com/

1. **推理过程**

最简单的推理过程是直接根据构建好的问题和模式生成 SQL 语句，这在某种程度上高估了通用大语言模型在专业领域的能力。就像人们倾向于将一个复杂任务分解为几个简单的子任务或步骤一样，基于提示工程的方法通常会设计特定的推理过程，以便使用大语言模型生成对查询的回复。Text-to-SQL 任务中的推理过程可以根据不同的推理模式进行分类。

- **思维链**：思维链（CoT）[38]是如今大语言模型中主要的推理策略，通常以“Let’s think step by step”这样的短语开头，以引发链式思考。这种方法特别适合处理自然语言问题中的复杂逻辑任务。DIN-SQL[11] 针对其复杂类别的问题，采用了人为设计的思维链步骤。对于非嵌套的复杂问题，思维链步骤包括模式关联和单一的中间表示。相反，对于嵌套的复杂问题，思维链步骤涵盖了几个子问题以及原始问题的相应子查询。
- **从少到多**：从少到多（Least-to-Most）[39] 是 Text-to-SQL 推理过程设计中另一种广泛使用的策略，在与大语言模型的一次交互中，它将一个复杂问题分解为一系列更简单的子问题。LTMP-DA-GP[62] 采用从少到多的方法分解自然语言查询，将 NatSQL[63] 映射到分解结果，并从 NatSQL 生成 SQL 语句。
- **分解**：除了上述推理策略之外，还存在一种简单但有效的推理过程，它主要将生成任务分解为与大语言模型的自定义交互，并采用各种技术来应对每个阶段的挑战。QDecomp[15] 首次提出了这种方法，它没有使用思维链或从少到多的策略，而是指示大语言模型以简化和迭代的方式将原始复杂问题分解为推理步骤。DIN-SQL[11] 将 SQL 语句分为三个复杂程度级别：“简单”、“嵌套复杂”和“非嵌套复杂”。MAC-SQL[13] 引入了一个多智能体框架，包括选择器、分解器和优化器。分解器智能体在预测最终的 SQL 语句之前，会生成一系列中间步骤（即子问题和 SQL 语句）。DEA-SQL[56] 遵循信息确定、分类与提示、SQL 生成、自我校正和主动学习的全局分解步骤来完成 Text-to-SQL 任务。
- **其他**：除上述策略外，还有一些专门设计的推理过程。BINDER[64] 首先利用大语言模型生成其特定领域语言 BINDER-SQL，这种语言与 SQL 语言一致，但一些列名和值被替换为 API 表达式，对应于子问题和原始表中的一些信息。然后，BINDER 再次使用大语言模型，根据 API 调用解决的子问题，将 BINDER-SQL 转换为 SQL 语句。R3[49] 针对 Text-to-SQL 任务提出了一种基于共识的多智能体系统。该系统由一个 SQL 编写智能体和几个不同角色的评审智能体组成。SQL 编写智能体和评审智能体之间经过几轮协商后，会达成共识并确定最终答案。

2. 示例选择

在基于大语言模型的 Text-to-SQL 方法的推理中，常常会通过纳入一些示例来提升 SQL 生成的性能。根据是否添加示例，这些方法可分为两类：零样本方法和少样本方法。基于大语言模型的零样本 Text-to-SQL 方法的例子包括 C3[10]、ReBoost[54]、Generic[55]、SGU-SQL[65] 以及 SQLfuse[60]。尽管零样本方法具有节省大语言模型标记数量的优势，但 Text-to-SQL 是一项复杂的任务，仅仅修改指令并不能有效地解决这一复杂任务。与零样本方法不同，少样本方法能够从示例中学习任务模式，而不单纯依赖于模型的预训练知识。这一特点提升了大语言模型的性能和适应性。对近期基于提示工程的 Text-to-SQL 方法中所使用的示例进行全面研究后，本文将示例主要分为分为两类，具体如下：

- **学习替代任务：**如前所述，为了处理复杂的 Text-to-SQL 任务，相关研究探索了各种推理过程，这些流程包含特定的步骤，或者涉及到之前不存在的新定义的子任务。例如，当为非嵌套的复杂问题生成 SQL 语句时，DIN-SQL[11] 遵循思维链风格的推理过程，并带有精心设计的步骤。它规定首先生成与问题对应的 NatSQL[63] 作为中间表示。随后，基于该 NatSQL[63] 生成最终结果。另一个例子是 BINDER[64]，它用语言模型 API 调用函数扩展了编程语言（如 SQL），并使用大语言模型将自然语言查询转换为扩展的 SQL 语言。然而，从模型的角度来看，DIN-SQL[11] 设计的步骤和 BINDER[64] 的转换过程都是陌生的，并且不容易通过人类语言提示进行描述。在这种情况下，它们都利用了大语言模型的上下文学习能力，并提供了一些示例。这极大地简化了提示中的指令设计。Divide-and-QDecomp[15]、Prompt[66] 和 COE-SQL[67] 也采用了这一思路。
- **学习 SQL 编写风格：**合适的示例能够显著提升大语言模型的性能[36]。同时，研究发现大语言模型对样本选择非常敏感，选择不合适的样本甚至可能产生负面影响。为了实现性能最大化，相关研究[12,68-69] 探索了如何选择合适的示例。最简单的方法是检索问题语义相似的示例。然而，即使不同数据库模式下的问题潜在意图相似且相应的 SQL 查询也有相似之处，但这些问题本身可能差异很大。为了弥合这一差距，像 Retrieval & Revision[69]、DAIL-SQL[12]、DEA-SQL[56] 和 PURPLE[70] 等研究首先会屏蔽原始问题中特定领域的词汇，以获取查询的框架。然后，它们检索问题框架语义相似的示例。其中，PURPLE[70] 进一步设计了四个层次的 SQL 框架抽象，并专注于更粗粒度的检索。一些研究[12,57,69] 尝试使用更多信息来检索示例。Retrieval & Revision[69] 通过向大语言模型提供指令来简化自然语言问题，并利用原始问题和简化后的问题来检索示例，这避免了因不常见的提问风格而带来的困扰，并增加了语料库中语法和措辞的多样性。DAIL-SQL[12]

利用自然语言问题和相应的 SQL 查询来检索示例，而 Open-SQL[57] 则利用自然语言问题、数据库模式和相应的 SQL 查询来检索示例。

2.2.3　后处理

为了进一步提升基于大语言模型的 Text-to-SQL 方法的性能和稳定性，相关研究对生成的 SQL 语句进行了后处理。本文将这些方法归纳为两类：自我校正和一致性处理。

1. 自我校正

在大语言模型生成一个答案之后，自我校正方法会利用特定问题和任务下的规则，让大语言模型检查答案的正确性。在 Text-to-SQL 的场景中，自我校正方法可以使用与 SQL 相关的规则进行检查，也可以将运行 SQL 语句所生成的结果或错误日志提供给大语言模型进行检查。Generic[55] 注意到了表值中额外空格的细微差异，并让大语言模型重新检查这些差异。而研究[68-69,71] 中的方法是，将原始提示词中过滤后的模式信息替换为完整的模式信息，并且如果大语言模型生成的 SQL 语句无法运行，则重新生成 SQL 语句。在研究[11,13,54,61,69] 中，不正确的 SQL 语句以及指示执行错误的信息将被用作提示，让大语言模型重新生成 SQL 语句。DEA-SQL[56] 分析了字段匹配和 SQL 语法方面的几个重要错误点，然后设计了特定的提示来纠正这些错误。研究[52] 为模型设计了单元测试、代码解释以及执行结果，以优化模型的输出。SQLfuse[60] 提出了 SQL 评判模块，并采用了少样本上下文学习策略，该策略利用来自外部 SQL 知识库的示例，并辅以事后反馈来丰富内容。

2. 自一致性处理

自我一致性方法[48] 主要采用多数表决策略，通过设置一定的温度参数，让同一个大语言模型针对同一个问题生成多个答案。然后，大语言模型选择出现频率最高的答案作为最终答案。在研究[10,12,57,61,64,72] 中，直接使用自我一致性方法对其大语言模型生成的 SQL 语句进行多数表决，取得了良好的性能提升效果。研究[73] 提出对多个答案进行重新排序，并训练一个验证器来检查生成的代码。PURPLE[70] 则基于生成的 SQL 语句的执行结果进行多数表决。与自我一致性方法不同，交叉一致性方法使用多个不同的大语言模型或智能体分别生成答案，或者检查 SQL 语句的有效性。PET-SQL[50] 提出了这种方法，它指示几个在较低温度参数下的大语言模型生成 SQL 语句，然后根据这些 SQL 语句的执行结果进行表决。为了发挥各自的优势，CHESS[61] 在其后处理阶段依次使用了自我校正和自我一致性方法。R3[49] 采用了多智能体框架，以循环的方式让具有不同专业知识的智能体针对 SQL 语句提供建议，这类似于交叉一致性方法和自我校正方法的结合。

2.2.4　相关工作局限性

基于提示工程的方法是当今 Text-to-SQL 任务的主流研究方向。相较于微调，基于提示工程的方法开销更小，在不同域之间的可移植性也更强。在预处理、推理、以及后处理等多个阶段都有相应的提升空间。当前的研究主要集中在少样本的示例选择、复杂问题的推理可靠性、以及 SQL 语法纠错等几个热门方向。同时，随着框架的不断细化拆分，端到端的延迟也随之不断增长。如何在关键的子任务改进和框架整体的时间复杂度、标记开销之间找到平衡，也是将来研究必须考虑的要素。本文第三章的研究为提示工程设计了整体的流水线框架，选择关键的子任务优化方法，试图逼近这一任务中大语言模型推理准确率的上界。

2.3　基于微调的 Text-to-SQL

大语言模型通过诸如检索增强生成（RAG）、上下文学习（ICL）和思维链（CoT）[38] 等提示方法，在 Text-to-SQL 任务上展现出了卓越的性能。然而，这些方法在很大程度上依赖于像 GPT-4[74] 这样强大的闭源大语言模型，需要将数据传输给闭源模型提供商，这可能会引发隐私问题[18,75]。与 GPT-4[74] 这样的大型闭源大语言模型相比，开源大语言模型在推理能力和遵循指令的能力方面存在一定的局限性。这是由多种因素造成的，包括模型规模较小、预训练语料库在数量上较少且质量欠佳，以及预训练的投入不够广泛等。此外，与 SQL 相关的内容在其整个预训练语料库中通常只占极小的比例[18]，这可能导致应用提示工程的方法时表现不佳。因此，开源大语言模型，特别是参数量级较小的模型，在 Text-to-SQL 任务中的主流应用方式是微调。

在目前的流程中，微调是指选取一个基础大语言模型，例如 Code-Llama[76]，并在特定场景中利用 Text-to-SQL 的数据集，针对与 Text-to-SQL 相关的任务（通常是 SQL 生成任务）进行进一步的训练。基于大语言模型的 Text-to-SQL 任务中的微调可以用公式 2.2 来表示。给定一个基础模型 M 和一个微调数据集 $\tau = \{(q_i, d_i, gt_i)\}$，其中 q_i 表示用户查询，d_i 是相应的数据库信息，gt_i 是真实的 SQL 查询语句。目标是通过损失函数 $Loss$ 来最小化预测的 SQL 查询语句 $M(q_i, d_i)$ 与真实的 SQL 查询语句 gt_i 之间的差异。

$$min_M = \sum_{i=0}^{|\tau|} Loss(M(q_i, d_i), gt_i) \tag{2.2}$$

2.3.1　训练目标

通常情况下，在基于大语言模型的 Text-to-SQL 任务中，微调的目标是生成 SQL 语句。也就是说，给定以特定设计风格呈现的问题表示和数据库信息，模型尝

试推断出正确的 SQL 查询语句。这类方法的示例包括 Open-SQL[57]、CodeS[18]、DAIL-SQL[12]、SQL-Palm[72]、Dubo-SQL[77] 以及 FinSQL[78]。除了上述生成 SQL 语句的目标之外，一些研究[13,60,79-80] 利用微调来提升基于大语言模型的 Text-to-SQL 推理过程中不同步骤的性能。MAC-SQL[13] 设计了一种三智能体架构，其中包括一个处理模式关联的选择器、一个负责问题分解和 SQL 生成的分解器，以及一个处理 SQL 校正的优化器。除了使用 GPT-4[74] 来充当这三个智能体之外，MAC-SQL[13] 还探索了使用经过微调的大语言模型来扮演这三个智能体的角色。DTS-SQL[79] 和 SQLfuse[60] 都探索了对大语言模型进行微调，以用于模式关联和 SQL 生成。DELLM[80] 创造性地对一个大语言模型进行了微调，使其能够基于给定的问题和数据库信息生成领域知识。然后，这些生成的领域知识将被整合到提示中，以辅助进行知识密集型的 SQL 生成。

2.3.2　训练方法

在大语言模型的微调中，存在两种主流的训练方法，即全量微调（FFT）和参数高效微调（PEFT）[41,81-84]。全量微调将大语言模型中的所有参数都视为可训练参数，而参数高效微调则只调整少量参数，这提高了训练效率并降低了训练成本。此外，研究[85]表明，在使用过程中，参数高效微调相比全量微调更不容易出现灾难性遗忘的情况，这再次证明了它的优越性。

在基于大语言模型的 Text-to-SQL 任务的背景下，微调既采用了全量微调，也采用了参数高效微调。在参数高效微调方法中，低秩适应（LoRA）[41] 和量化低秩适应（QLoRA）[82] 是最常用的方法。低秩适应（LoRA）[41] 是“Low-Rank Adaptation”的缩写，它冻结预训练模型的权重，并将可训练的低秩分解矩阵注入到 Transformer 架构的每一层中。通过这种方式，低秩适应显著减少了用于下游任务的可训练参数的数量。量化低秩适应（QLoRA）[82] 是一种基于低秩适应开发的微调方法。通过引入 4 位 NormalFloat、双重量化和分页优化器等一系列创新举措对低秩适应方法进行了改进，在保持性能的同时减少了内存使用量。在基于大语言模型的 Text-to-SQL 任务中采用全量微调的例子有 MAC-SQL[13]、CodeS[18] 和 SQL-PaLM[72]，而采用低秩适应（LoRA）[41] 和量化低秩适应（QLoRA）[82] 的例子有 OpenSQL[57]、DELLM[80]、FinSQL[78] 以及 SQLfuse[60]。

2.3.3　训练数据

大多数基于微调的 Text-to-SQL 研究方案[12-13,57,72,77,80] 直接采用了开源 Text-to-SQL 基准测试的训练集。其中，DTS-SQL[79]、DAIL-SQL[12] 和 SQL-PaLM[72] 使用 Spider[8] 训练集对大语言模型进行微调。OpenSQL[57]、DELLM[80] 和 Dubo-SQL[77] 使用 BIRD[33] 训练集。MAC-SQL[13] 则同时使用这两个数据集

进行实验。训练数据可能会经过预处理（例如模式关联和知识补充），转换为设计好的表示格式。

CodeS[18] 为如何构建微调数据集以提升通用 Text-to-SQL 能力，以及如何构建微调数据集以自适应迁移到新领域的数据库提供了思路。为提升通用 Text-to-SQL 能力，CodeS 收集了 21.5GB 的数据，包括 SQL 相关数据（11GB）、自然语言到代码的数据（6GB）和自然语言相关数据（4.5GB），旨在同时提升 SQL 生成能力和自然语言理解能力。为自适应迁移到新领域的数据库，CodeS[18] 收集了一些真实用户查询，手动标注相应的 SQL 查询，并利用 GPT-3.5 以少样本的方式生成更多的 {问题，SQL} 对，确保以用户为导向的真实性。同时，CodeS 利用 Text-to-SQL 基准测试中的 SQL 模板及其问题模板，将新领域数据库中的表、列和值填入模板，生成了多样化的 {问题，SQL} 对。

2.3.4　相关工作局限性

出于对企业隐私信息和数据安全的考虑，工业领域的 Text-to-SQL 任务更适合采用微调的方法。然而，与提示工程方法相比，微调方法在研究界尚未得到充分的探索。首先，由于 GPT 系列等闭源模型[74] 具有出色的性能，大多数基于闭源模型的方法能够取得更好的效果。再加上应用程序编程接口（API）调用的成本较低，对开源模型微调的研究没有像提示工程方法那样受到广泛关注，这在一定程度上导致了开源模型微调方法的研究相对滞后。其次，提示工程方法在算法层面有更多的创新点，相比之下，微调方法在算法层面的创新点较少，更多地依赖于训练技术、基础模型以及训练数据质量等方面的进展。这也导致了基于微调的 Text-to-SQL 方法受关注度较低，研究进展较为缓慢。相较于这些工作，本文第四章的工作集中在数据增强和多任务协作，尝试为轻量化开源模型找到可以与大型闭源模型性能相当的训练方法。

2.4　表格问答算法

表格问答是常见的表格理解任务中最具挑战性的任务之一，也是目前和现实生活联系最紧密、应用最广泛的研究任务之一。表格问答任务要求模型同时理解自然语言问题和相应的表格数据，而表格数据又与 Text-to-SQL 任务中的数据库和结构化表格数据息息相关，所以这二者通常被置于同一框架中考量，二者的方法也经常可以互为借鉴。本节也将按照 Text-to-SQL 任务的叙述思路，介绍表格问答的发展现状和未来趋势。

2.4.1　基于提示工程的表格问答算法

与 Text-to-SQL 不同，表格问答往往需要在单元格值，而非列名中检索关键证据，这意味着其输入大语言模型的序列往往随规模扩大呈指数级别增长。同时，表格数据本质上是一种独特的二维结构，这与大语言模型预训练语料库中占主导地位的线性文本形成对比。大多数模型并未针对处理表格数据进行专门优化，在完全理解表格和有效利用表格数据进行推理的能力方面存在显著差距。

基于提示工程的表格问答算法主要研究提高表格推理的性能，Mix-SC[28]通过为每个问题生成多个候选答案来探索大语言模型的随机性。它采用了两种提示技术：一种是引导大语言模型充当 Python 代理来执行脚本，另一种是利用思维链（CoT）提示[38]来引导逐步推理。最终答案通过自一致性机制选择，该机制选择出现频率最高的答案。也有一些方法将程序生成和执行与大语言模型提示相结合，Binder[64]首先促使大语言模型根据输入问题生成一个初始程序，然后通过与大语言模型的迭代交互来识别和优化具有挑战性的部分。最终优化后的程序由解释器执行以生成表格问答的答案。类似地，TabLaP[86]使用大语言模型来生成针对数值推理的 Python 脚本。尽管基于大语言模型的表格问答模型具有很强的推理能力，但它们在处理复杂表格结构时存在困难，并且对噪声非常敏感，这限制了它们在大型自由格式表格上的有效性。

研究表明，当为表格问答模型提供一个仅包含与问题相关信息的子表格（而非完整表格）时，模型的性能会显著提高[87-89]。现有的提取子表格方法同样可分为两类:

- **软分解**：这些方法在弱化无关信息的同时强调关键内容，使模型能够专注于相关数据，而无需显式删除无关条目。例如，CABINET[89]提出了一个相关性评分器和一个相关单元格预测器，用于为表格单元格分配权重。这些方法依赖于模型微调，这限制了它们的模型通用性，并且通常受限于固定的输入标记长度，从而限制了可扩展性。
- **硬分解**：这些方法删除无关内容，提取一个子表格作为表格问答模型的最终输入表格。例如，Chain-of-Table[90]扩展了思维链（CoT）框架。它使用大语言模型在每一步逐步提取一个子表格。DATER[87]也利用了思维链的思想，将表格和输入问题都分解为结构化的子表格和子问题。Learn-to-Reduce[88]引入了一个列选择器和一个行选择器，它们是经过微调的语言模型，仅提取与问题相关的行和列以形成子表格。这些方法往往忽略表格结构。它们将表格扁平化，仅根据语义相关性选择行/列，从而导致信息丢失。

相较于这些工作，本文第五章的工作将大语言模型驱动的推理与 Text-to-SQL 技术相结合，在保留表格问答关键信息的同时，实现了更具结构感知的表

格分解。值得说明的是，本文只检索关系密切的行、列值并输入常数项的单行表格信息，并未将完整表格纳入大语言模型输入，而上述工作都需要至少输入一次完整表。这导致在面对大型表格时，本文工作在成本开销方面占有显著的优势。

2.4.2　基于微调的表格问答算法

基于微调的工作主要针对预训练模型，一些工作针对表格数据进行结构调整，对 Transformer 架构[44]进行修改，以更好地捕捉表格结构。TaBERT[23]结合了垂直自注意力机制，在 BERT[4]的基础上统一文本和表格表示。TUTA[91]采用了基于层次树的注意力机制，结合列和表格段掩码来对表格表示进行建模。TAPAS[25]通过添加行、列和等级嵌入以及表格单元格掩码来扩展 BERT。

另一些工作通过端到端模型进行微调，TAPEX[92]使用从 WikiTableQuestions 数据集衍生的大型合成数据集对 BART[93]进行预训练。OmniTab[94]也使用 BART 作为其主干模型，同时在真实数据和合成数据上进行预训练，这些数据包括来自 Spider[8]数据集的 SQL 查询语句，以及通过 SQL-to-Text 模块从 SQL 查询生成的合成自然语言句子。与最新的大语言模型相比，预训练语言模型在模型规模较小的情况下，在理解表格语义方面的效果有限，导致在表格问答任务中的准确性不够理想。然而，现有的大语言模型并没有针对表格二维结构的专用架构，更缺少高质量的表格数据集。针对表格数据微调的大语言模型仍是当前研究中的空白。

2.4.3　相关工作局限性

大多数研究单纯集中于提升表格推理性能，它们在引用多种表格操作工具和推理验证工具的同时，不可避免地增长了整体工作流程的标记开销和时延。同时这些方法往往要求输入全部表格，可能会超过大语言模型的输入标记上限从而无法适用。本文的研究针对大型表格难以直接推理的问题，利用现有的 Text-to-SQL 任务生成的 SQL 语句作为工具，检索表格相关信息，排除噪声，填补大型表格推理问答任务的空白。

2.5　本章小结

Text-to-SQL 任务在大模型时代迎来了新的发展，从提示工程和微调两条技术路线出发，适用于不同场景衍生出了大量研究。在接下来的章节中，本文将围绕这两条技术路线中的核心问题展开讨论和实验，改进方法提升模型性能，并延伸出将这一任务泛化于表格问答任务的方案，为企业的信息化和智能化转型贡献自己的力量。

第 3 章　基于多智能体协作和模式对齐的 Text-to-SQL 算法

本章主要阐述在提示工程这一技术路线中提出的 Text-to-SQL 算法 SA-SQL。具体地，本章 3.1 节首先介绍研究的挑战和动机，3.2 节对 Text-to-SQL 进行了形式化定义，并且介绍了算法涉及的智能体、模式对齐和关系代数的相关概念，3.3 节对提出的方法进行了整体性描述，同时详细介绍了各个智能体的实施细节，3.4 节在两个真实世界数据集对 SA-SQL 和基线方案进行了一系列实验和分析，验证了 SA-SQL 的有效性，最后 3.5 节对本章进行了总结。

3.1　引言

Text-to-SQL 任务旨在将自然语言查询自动转换为结构化查询语言（SQL）。该任务能够提升数据库访问能力，使人们无需掌握 SQL 知识即可操作数据库。近年来，随着大语言模型的发展，一个重要的方向是通过思维链（CoT）[66]、智能体[13]和上下文学习[11]等方法完成该任务。大多数研究[10-14]聚焦 Text-to-SQL 局部任务的优化，进行提示词编写和智能体构造，最后再通过一系列任务的协作优化 SQL 的生成过程。尽管这种大语言模型驱动提示工程的方法显著提升了 Text-to-SQL 任务的能力上限，现有的方法仍有相当的局限性，具体如下：

- **整体框架模糊**：多数方法都未涉及到 SQL 生成的全部过程，这使得这些方法未能充分发挥潜力。例如，一些研究缺乏对数据库中存储信息的验证，另外一些对生成结果缺少纠错机制，或者存在少样本学习缺失等问题。
- **幻觉累积**：由于大语言模型的不稳定性以及智能体之间缺乏连贯性和耦合性，后执行的智能体可能不会使用或仅部分使用先前运行智能体的输出，这导致幻觉累积和性能损失。
- **复杂推理困难**：自然语言与 SQL 的语法结构存在差异，这使得复杂问题到 SQL 的转换愈加困难。在大语言模型出现之前，研究者们开发了使用中间语言生成 SQL 的方法来缓解这个问题，但目前关于如何构建指令提升大模型推理阶段准确率的研究仍然不足。

为解决上述问题，本章首先面向 SQL 构建的过程，定义了一个标准的 Text-to-SQL 框架，该框架涵盖了当前大语言模型驱动的 Text-to-SQL 任务的研究。一个完整的 Text-to-SQL 任务框架应包括三个模块：预处理、推理和后处理：

- **预处理**：处理和构建所有与自然语言查询无关的辅助信息，包括数据库模式信息、数据库嵌入、少样本库等。然后从数据库中提取和选择生成 SQL 所需的元素，包括子自然语言查询、少样本示例，以及表、列、值等。缺

少这一步骤可能会导致大语言模型无法将自然语言问题中的实体与数据库中的模式建立起映射关系。

- **推理**：根据 SQL 语法、少样本示例和准备好的信息，将自然语言查询转换为 SQL。现有的方案大多是单步生成最终结果，在面对涉及到复杂嵌套或聚合计算的问题时极易产生错误。
- **后处理**：根据执行结果，使用对齐策略和规则检查并优化 SQL。然后，基于自一致性投票结果选择最终的 SQL。缺少这一步骤可能会导致生成的 SQL 无法与数据库中单元格值或某些特定的语法结构对齐，有时即使语义正确，也无法在数据库查询中得到理想的结果。

接下来，本章工作分析了多智能体协作驱动的 Text-to-SQL 任务中常见的幻觉现象，并针对这些幻觉设计了模式对齐策略。通过确保特定智能体的功能、输入和输出的一致性，减少模型生成过程中的幻觉。这种对齐方法被用于预处理与推理阶段之后，负责对智能体的输出进行有针对性的调整。最后，针对复杂问题推理困难的挑战，本文设计了一种改进思维链策略，将关系代数作为中间语言引入推理过程，在每一个子步骤同时输出自然语言表示的子问题以及相应的关系代数。这一策略解决了传统思维链流程的不可控性，优化了问题分解的粒度使其更加适用于 SQL 生成场景，从而降低复杂问题的推理难度。实验结果表明，SA-SQL 在两个真实世界数据集中均取得了优秀的性能表现，为大语言模型在复杂任务场景中的可用性提供有力支撑。

3.2　预备知识

3.2.1　问题定义

给定一个 Text-to-SQL 的指令数据集 $D = (d_i, q_i, s_i)_{i=1}^{N}$，其中 d_i 是一个 SQL 数据库，q_i 是一个可能带有问题提示的自然语言问题，s_i 是一个真实的 SQL 查询。Text-to-SQL 的目的是利用大语言模型，基于由 d_i 和 q_i 构建的提示生成一个 SQL 查询 s_i^*，且预测的 SQL 查询的执行结果与真实的 SQL 查询 s_i 的执行结果一致。

3.2.2　智能体

智能体（Agent）的实现主要依赖大语言模型固有的指令跟随以及上下文学习能力。在其典型应用中，AutoGPT[95] 采用单智能体范式，为 AI 模型扩展了许多实用工具；而 AutoGen[96] 是一个开源框架，使开发人员能够构建可定制的对话式智能体，这些智能体可以在多种模式下运行，通过结合大语言模型、人类输入和工具来执行任务。本文旨在通过利用专注于 Text-to-SQL 不同环节的智能体

的优势，提高现实场景中 SQL 查询解析的准确性和效率。

为强化智能体在特定任务场景中的能力，本文在推理智能体，以及后处理阶段中的自纠正智能体中引入了少样本示例学习。前者旨在规范推理智能体的推理路径，并对关系代数先验知识作一定补充。示例根据掩码问题相似性[68]动态选择，以确保上下文中提取到的关系代数操作能为复杂推理提供更多帮助；对于后者，则需要事先针对错误类型准备不同的少样本，使大语言模型在优化阶段能够更清楚地了解如何根据不同的错误纠正 SQL。错误纠正示例的构建需要标注出原始 SQL 中的具体错误和纠正建议，并与错误类型相对应。

3.2.3　模式对齐

幻觉（Hallucination）是影响大语言模型可用性的关键问题，在 Text-to-SQL 任务中也同样存在。此外，多智能体协作中产生的累积错误加剧了大语言模型的幻觉问题。例如，如果负责筛选功能的智能体以某种方式从数据库中选择列，并且由于模型幻觉生成了错误的列名，那么这个错误将在后续的生成阶段持续存在，因为后续阶段缺乏对正确数据库结构的了解。幻觉在多智能体大语言模型工作流程中很难自行消失，其总量几乎单调不减。

在 Text-to-SQL 任务中，常见的幻觉现象主要指未能遵循指令和输出结果不稳定。具体表现为：生成不存在的列、更改数据库列名、句法错误、数据库值与列不匹配以及未能遵守提示中设置的规则。受研究[97]启发，本文在预处理、推理智能体输出结果后进行模式对齐，确保各个智能体的功能实现逻辑一致性，并将对齐结果传递给下游智能体。同时推理智能体的思维链策略要求在分解子问题的同时输出相应的关系代数式，以关系代数的语法为基础约束问题分解的语义空间。这种机制有效地扩展了多个智能体之间的协作链，同时最大限度地减少了幻觉的引入。

3.2.4　关系代数

SQL 与自然语言表示之间存在语法的差异。在通过分解问题进行复杂 SQL 推理时，尽管现有的思维链方法可以有效地将问题分解为若干子问题，但在将子问题映射到相应的 SQL 子句时却面临困难。一个完整 SQL 查询包含若干必要的子句，更适合作为整体操作流程的声明。这极大限制了问题分解的灵活性，也是以往分解工作[15]未能显著提升 Text-to-SQL 性能的主要原因。

本文选择关系代数（Relational Algebra）作为中间表示，其设计初衷是描述关系型数据库操作，其基本定义如表 3.1 所示。关系代数包含两个基本操作：投影（Project）和选择（Select）。与 SQL 中的 `SELECT` 和 `FROM...WHERE` 相比，它们具有更明确的作用域和函数定义。此外，关系代数与 SQL 的一个重要区别

在于：每个关系代数表示都是有效的可执行操作符，输入一组关系并返回处理后的关系。相反，SQL 作为查询语言为保持语义完整性，牺牲了表示某些自然语言问题的能力。如表 3.2所示，一个 SQL 查询可以视为若干关系代数操作的集合，而一个关系代数却只能映射到 SQL 查询的子句。二者虽然可以互相转换，但关系代数粒度更细，更适合作为问题分解的中间表示。本文设计了一个基于提示的智能体，将复杂问题分解为自然语言描述的子问题及对应的关系代数表示。关系代数的特性确保推理中的每一个子问题都可以映射到相应的数据库操作。

表 3.1　常见关系代数操作符

操作符	表示符号	描述
Select	$\sigma_{predicate}(R)$	选择满足给定谓词的元组
Project	$\pi_{columnmn}(R)$	返回仅包含原始关系中某些属性（列）的关系
Intersect	$R_1 \cap R_2$	返回同时存在于两个原始关系中的关系
Union	$R_1 \cup R_2$	返回合并两个关系的元组且无重复项的关系
Set Difference	$R_1 - R_2$	返回所有在 R1 中而不在 R2 中元组的关系
Join	$R_1 \bowtie_\theta R_2$	返回基于给定连接条件合并而成的关系
Aggregate	$\gamma_{column}(R)$	根据分组条件将元组流聚合为分组流

表 3.2　关系代数和 SQL 子句之间的映射关系

关系代数表达式	SQL 子句表达
$\sigma_{predicate}(R)$, R 为正常关系	< > `FROM` R `WHERE` predicate
$\sigma_{predicate}(R)$, R 为聚合生成	< > `HAVING` predicate
$\pi_{columnmn}(R)$	`SELECT` column
$R_1 \cap R_2$	R1 `INTERSECT` R2
$R_1 \cup R_2$	R1 `UNION` R2
$R_1 - R_2$	R1 `EXCEPT` R2
$R_1 \bowtie_\theta R_2$	< > `FROM` R1 `JOIN` R2 (`ON` R1.column1 = R2.column2)
$\gamma_{column}(R)$	< > `GROUP BY` column

3.3　算法框架

本节将详细介绍 SA-SQL 的具体内容。为确保解释清晰，接下来的说明会将各模块与其相应的对齐策略相结合，并按照它们在 SA-SQL 框架中的运行顺序进行描述，整体框架如图 3.1。

3.3.1　预处理

在预处理阶段，SA-SQL 首先需要构建有效的数据库提示，其具体过程如下。此外，少样本示例的构建也在这个阶段完成，包括添加思维链信息和为不同错误类型设计的自纠正少样本示例。

1. 模式链接

在现实场景中，数据库通常包含大量表和列，导致数据库提示极长。当这些提示超过语言模型的最大上下文长度时，必须进行截断处理，但这可能会遗漏生

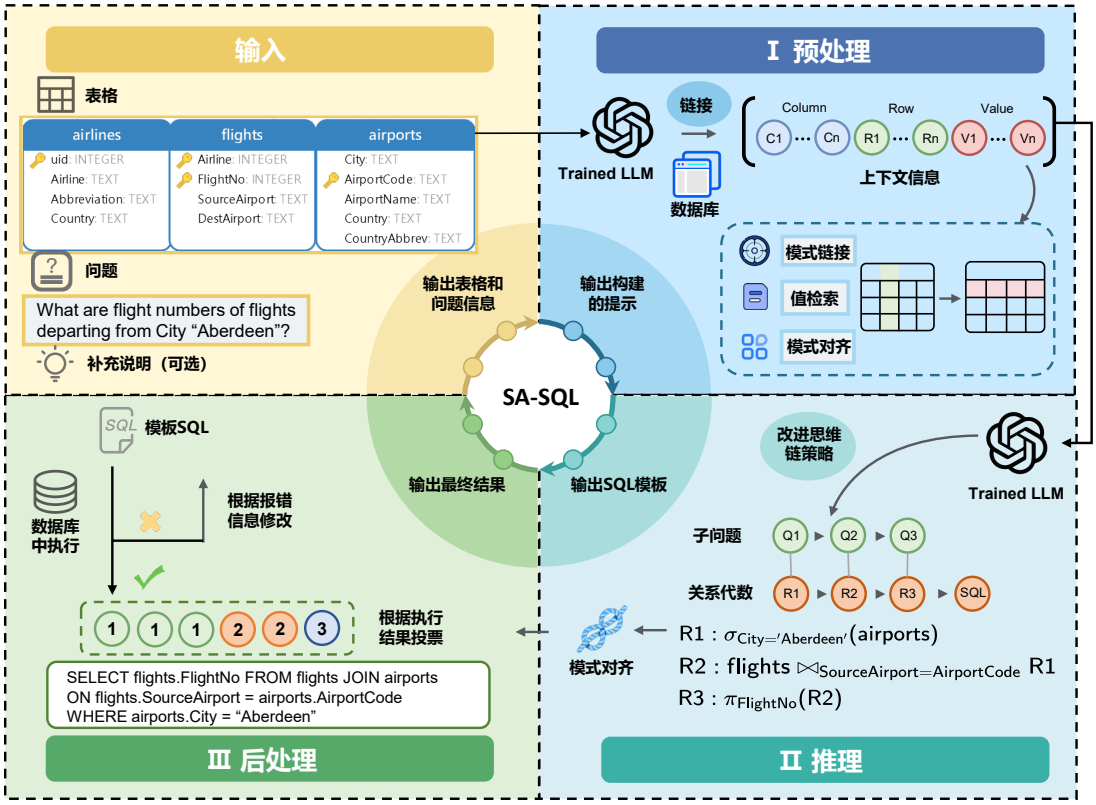

图 3.1　SA-SQL 整体框架

成目标 SQL 查询所需的关键表或列。因此，必须采用一种在不牺牲关键模式信息的前提下最小化数据库提示长度的方法。本文参考工作[10]中的方法，使用模式链接智能体为每个 Text-to-SQL 样本保留数据库中最相关的表和列。具体来说，给定一个数据库和一个问题，智能体会根据用户问题预测表和列的相关度分数，保留前 top_{k1} 个表和每个表的前 top_{k2} 个列。这一智能体不仅缩短了数据库提示的长度，还减轻了后续推理过程的选择压力。

2. 值检索

从数据库中提取与问题对齐的值可以帮助语言模型在生成谓词时更好地执行模式链接。对于 Spider 数据集中的自然语言问题“What are flight numbers of flights departing from Aberdeen”，数据库 `airports` 表中的 `City` 列的值为 `Aberdeen`。因此，将信息 `airports.City = 'Aberdeen'` 集成到数据库提示中可以指导模型生成准确的 SQL 查询谓词。为计算数据库值与问题的匹配程度，本文使用了一种“粗到细”的匹配方法。这种方法的本质在于利用索引进行快速但粗略的初始搜索，然后使用最长公共子串算法（LCS）算法进行细粒度的匹配过程。具体来说，首先构建 BM25（Best Matching 25，一种高效的文本检索算法）索引，以便为存储在每个数据库中的所有值提供支持。当接收到用户的问题时，索引首先基于问题从整个数据库中提取数百个潜在相关值。然后，使用 LCS 算法计算问题与这些潜在相关值之间的匹配程度，以找到最相关的值。

3. **提示构建**

在最终构建的 SQL 生成提示中，本文还添加了如下信息：

- 列数据类型：它决定了其验证规则和允许的操作。例如，像 INTEGER 和 REAL 这样的数值类型支持算术操作，而字符串类型不支持。如果某些数据以字符串类型存储，则在对其执行算术操作之前，必须在 SQL 查询中使用 CAST 函数。
- 补充说明：数据库模式中的歧义在现实世界的数据库中很普遍，由于大语言模型通常使用语义相似性进行模式链接，这可能导致模型选择错误的表或列用于生成 SQL 查询。将数据集提供的补充说明纳入模式链接的输入和最终提示中，可以帮助大语言模型准确建立问题实体与数据库列的关系。
- 代表性值：除了列名，代表性的单元格值也很有益。例如，为了生成 `student.gender = 'F'` 这样的谓词，必须告知大语言模型 `gender` 列提供两个值：`M` 和 `F`。同样，对于像 `SUBSTR(birthday, 1, 4) = '2009'` 这样的谓词，模型应该知道 `hiredate` 列的具体格式是 `YYYY-MM-DD`。为了解决这一问题，本文使用 SQL 查询 `SELECT DISTINCT {COLUMN} FROM {TABLE} WHERE {COLUMN} IS NOT NULL LIMIT 2` 来捕捉每一列的两个独特的非空代表性值。通过提供这些有洞察力的值，大语言模型更有可能生成准确且与上下文相关的 SQL 查询。
- 主键与外键：主键唯一标识表中的每一行，而外键则在表之间创建关联。纳入主键和外键信息可以指导模型推断出适当的连接路径，确保准确生成 JOIN ON 子句。在实践中，本文为每个主键列使用唯一标识符，并在数据库提示中将每个外键表示为 `{TABLE1}.{COLUMN1} = {TABLE2}.{COLUMN2}`。

4. **模式对齐**

在预处理阶段结束时，本文使用模式对齐策略检查预处理的结果，其要求从自然语言查询中提取问题的目标实体，并在生成的候选列中找到对应列，这有助于最终 SELECT 子句中内容的准确性。为避免因不同表中同名列导致的误选，本文通过将每个表的主键以及所有与候选列同名的列重新整合到最终结果中来扩展模式信息，从而保证所选列的准确性和全面性。最后，所有提取的信息和对齐的内容都将输入到推理阶段。

3.3.2　推理

本阶段的任务是使用思维链提示策略驱动大语言模型生成推理子步骤并完成 SQL 语句。

1. 思维链策略改进

本文设计了一个推理智能体，将复杂问题分解为自然语言描述的子问题及对应的关系代数表示。具体而言，这是一种定制的思维链方法，相较于传统的思维链策略只有自然语言的子步骤，本文提出的策略要求每一个子步骤都必须映射到相应的关系代数表示，以鼓励大语言模型更多地关注数据库模式内部的逻辑。以表 3.3为例，假设 `student` 表中存在甲、乙、丙 3 名学生，但 `enrollment` 表中只存在甲选择的一门数学课和乙选择的一门语文课，丙未曾选课。在查询“未选择数学课的学生”这一问题时，正确答案应包括乙和丙。加入关系约束后智能体可以将分解为 5 个可执行步骤，并注意到“非”这一逻辑关系对应的主体是“学生”，生成的 SQL 可以得出正确的答案。而传统的思维链方法只能将问题拆解成 2 个子问题，而这些子问题难以对 SQL 生成产生帮助，最后错误地在“选课”这一主体中使用了“非”的逻辑关系，忽视了未在“选课”中出现的丙。

本文提出的加入关系代数的渐进式生成方式强调大语言模型对生成的 SQL 结构的理解，同时也便于识别大语言模型推理过程中的错误。在构建少样本示例时，系统会根据掩码问题相似性检索最相似的前 K_m 个查询，并将它们相应的子问题和关系代数操作集合用作最终指令中的少样本示例。同时，相较于 Nat-SQL[63] 等中间语言表示（一般是忽略特定语法元素，如 JOIN 操作和函数格式），关系代数属于大语言模型的先验知识域，这使得本文不需要像 DIN-SQL[11] 一样，构建大量的上下文提示以供模型学习，降低了输入端的巨大消耗。针对同一问题的 SQL 生成将重复 K_v 次，用于后续的自一致性投票以减轻模型推理幻觉。

2. 模式对齐

SQL 查询生成过程中的错误主要源于语法、数据库、自然语言查询和数据集风格的差异。此外，使用大语言模型在较高温度下进行采样以获得多样化的答案，这一过程可能会引入错误和噪声。本阶段的对齐策略是减少由这些差异导致的偏差。具体而言由以下三个步骤组成：

- **值对齐**：确保数据库中的列和值在 SQL 中得到正确表示。如果存在不匹配的情况，则进行纠正。一个常见的例子是 SQL 查询与数据库中存储的字符串因大小写差异导致的查询失败。
- **函数对齐**：对 SQL 聚合函数进行标准化处理，以防止因表达式错误而引发的问题。包括处理不恰当的聚合函数、嵌套结构以及冗余的 JOIN 操作。
- **偏好对齐**：解决与数据集特性相关的问题，例如 IS NOT NULL 的使用，以及在 MAX 和 LIMIT 1 之间的选择。

完成上述步骤后，将执行 SQL 语句并进入后处理阶段。

表 3.3　不同思维链策略的样例分析

表格信息 student(id, name), 值 [(1,‘甲’),(2,‘乙’),(3,‘丙’)] enrollment(stuid, courseid), 值 [(1,1001),(2,1002)] course(id, type), 值 [(1001，‘Math’),(1002,‘English’)] **问题** Find the IDs of students who did not choose the math course.
常规思维链推理流程 1. 找到所有选择了数学课的学生 2. 从所有学生中排除这些学生 **生成 SQL** SELECT e.stuid FROM enrollment e WHERE e.stuid NOT IN (SELECT e.stuid FROM enrollment e JOIN course c ON e.courseid = c.id WHERE c.type = ‘Math’) **SQL 执行结果** **2**
思维链 + 关系代数推理流程 1. 选择数学课程 $\sigma_{coursename='Math'}(course)$ 2. 连接选课表 $enrollment \bowtie \sigma_{type='Math'}(course)$ 3. 投影学生 ID $\pi_{stuid}(enrollment \bowtie \sigma_{type='Math'}(course))$ 4. 获取所有学生 ID $\pi_{id}(student)$ 5. 求差集 $\pi_{id}(student) - \pi_{stuid}(enrollment \bowtie \sigma_{type='Math'}(course))$ **生成 SQL** SELECT id FROM student EXCEPT SELECT e.stuid FROM enrollment e JOIN course c ON e.courseid = c.id WHERE c.type = ‘Math’ **SQL 执行结果** **2 3**

3.3.3　后处理

后处理阶段的工作是对生成的 SQL 进行优化和筛选，包括根据执行结果纠正错误，以及从多个候选 SQL 中选出最佳答案。后处理过程包括以下两个步骤：

1. **自纠正**: 执行 SQL，并根据执行结果中的错误细节进行修复，例如语法错误或空结果。每种错误类型都对应不同的少样本示例和自纠正指令，同一问题示例数量为 K_c。这一步的目的是避免因小细节导致的无结果或执行失败等问题。
2. **自一致性投票**: 由于大语言模型生成结果的可变性，以及在复杂任务中指令遵循方面频繁出现的问题，本文采取复杂推理中常用的自一致性投票策略。排除无法修复和执行结果为空的 SQL，在执行结果中选择输出一致性最高的选项。此外，在答案相同的 SQL 查询中，本文选择执行时间最短的查询。这种方法可以缓解大模型推理的不稳定及幻觉现象，其中参与投票的候选结果数量为 K_v。

3.4　实验结果分析

3.4.1　实验设置

1. 数据集

本文采取目前 Text-to-SQL 任务中最通用的两个数据集作为评测标准，考虑到数据集的领域和 SQL 复杂度，BIRD[33]数据集的数据库类型相对较少，但数据库结构更复杂，SQL 的平均难度更高。相比之下，Spider[8] 数据集拥有丰富多样的数据库，但 SQL 的平均难度相对较低。两个数据集均采用验证集作为评测的基准。

- **BIRD** 是一个开创性的跨域数据集，用于研究广泛的数据库内容对 Text-to-SQL 解析的影响。BIRD 包含超过 12751 个独特的 {问题-SQL} 对，95 个大型数据库，总大小为 33.4GB。它涵盖了超过 37 个专业领域，如区块链、曲棍球、医疗保健和教育等。此外，BIRD 为特定样本提供外部知识，以便生成正确的 SQL 查询，本文将其与问题整合以生成提示。
- **Spider** 是一个由 11 名耶鲁大学学生标注的大规模复杂跨域语义解析和 Text-to-SQL 数据集，其目标是是开发跨域数据库的自然语言接口。它由 10181 个问题和 5693 个独特的复杂 SQL 查询组成，涉及 200 个包含多个表的数据库，涵盖 138 个不同领域。

2. 评估指标

遵循 BIRD 和 Spider 测试套件的评估标准，本文评估了两个指标：执行准确率（EX）和基于奖励的有效效率得分（VES）。EX 定义为预测的 SQL 与正确 SQL 执行结果相同的比例，而 VES 用于衡量预测的 SQL 在执行相同任务时的执行效率。

3. 基线模型

本文选择了 BIRD 和 Spider 排行榜中由大语言模型提示工程驱动的方法作为基线：

- **ChatGPT** 和 **GPT-4**[74]采用零样本 Text-to-SQL 提示来生成 SQL。
- **C3-SQL** [10]通过三个模块（清晰提示、带提示校准和一致输出）构建了一种系统的零样本 Text-to-SQL 方法。
- **DIN-SQL** [11]通过多个模块将问题分为不同类型，并通过各种提示指示大语言模型生成最终的 SQL。
- **DAIL-SQL** [12]通过为不同问题选择相似的问题-SQL 对作为少样本示例来辅助大语言模型生成 SQL。
- **MAC-SQL** [13]通过使用子数据库和子问题来简化大语言模型面临的挑战，从而生成 SQL。

- **MCS-SQL** [14]使用多个提示生成多组 SQL，并采用统一的多选选择来决定最终的 SQL。
- **CHESS** [61]通过构建高效的检索和模式修剪方法来减少冗余信息的干扰，从而提高 SQL 生成的有效性。

4. 实施细节

为了展示 SA-SQL 的能力，本文选择的大语言模型基座为用于生成的 GPT-4 和用于对比的 ChatGPT。为了提高效率和可比性，本文在 BIRD 验证集上进行了消融实验。这些消融实验的目的是评估 SA-SQL 在各种模型上的泛化能力，并验证该方法中每个智能体的有效性。在预处理阶段，本文将温度设置为 0，最大候选表数量 top_{k1} 为 3，最大候选列数量 top_{k2} 为 10。在推理和后处理阶段，温度设置为 0.7，推理少样本示例数 K_m 和自校正少样本示例 K_c 从 {0,1,2,3,4,5} 中选择，自一致性投票的数量 K_v 为 15。

3.4.2　主要结果

表 3.4　BIRD 和 Spider 数据集上不同方法的执行准确率和效率

方法	BIRD		Spider
	EX	**VES**	**EX**
ChatGPT	35.6	30.4	73.9
GPT-4	49.1	49.7	76.6
C3-SQL + ChatGPT	44.2	45.4	80.4
DIN-SQL + GPT-4	50.7	58.7	82.8
DAIL-SQL + GPT-4	54.3	56.0	83.6
MAC-SQL + GPT-4	57.6	57.8	86.8
MCS-SQL + GPT-4	63.4	61.2	89.5
CHESS	65.0	62.8	87.2
SA-SQL + GPT-4	**66.6**	**66.4**	**90.8**

BIRD 结果：本文在表 3.4中展示了基线模型和 SA-SQL 在 BIRD 数据集上的性能，其中最优方案加粗显示，次优方案以下划线显示。本文在 BIRD 验证集上的 EX 达到 66.6%，VES 达到 66.4%，超越了所有用于比较的基线模型。值得注意的是，所有基线模型均基于提示工程针对生成某一环节进行优化，这证明了本文工作统一范式进行全过程优化的必要性。

Spider 结果：为了展示 SA-SQL 的泛化能力，本文还评估了它在 Spider 数据集上的性能。SA-SQL 仍保留默认配置，除了为适应数据集格式进行必要的调整外，没有进行其他更改。表 3.4显示了各种方法的性能，本文的 EX 达到 90.8%。这表明了 SA-SQL 在优化 SQL 结构提高执行准确率的性能。

为评估 SA-SQL 在复杂推理场景中的表现，本文依据 Spider 和 BIRD 数据集的难度划分规则，比较了基座模型 GPT-4 和 SA-SQL 在不同难度级别问题上的性能表现，如图 3.2所示。Spider 将问题划分为简单、中等、困难、极难四个等

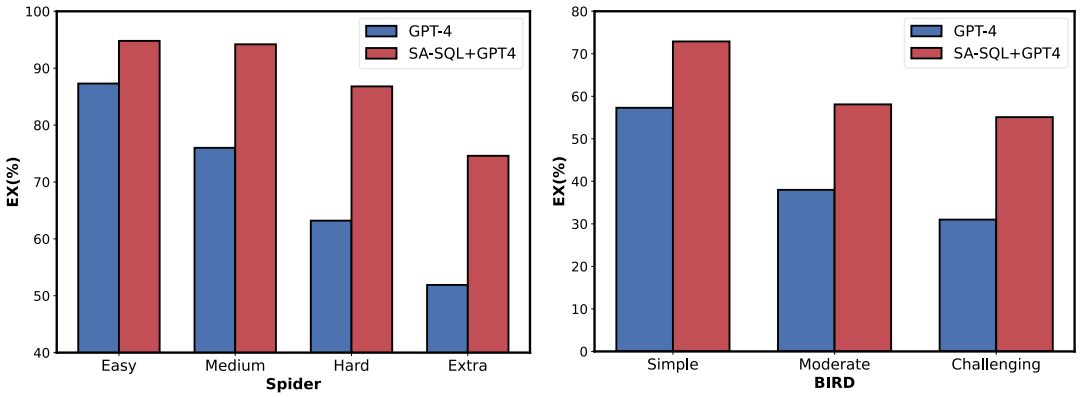

图 3.2　不同难度样例中执行准确率比较

级，SA-SQL 在“极难”类别最复杂的生成场景下取得了显著突破，而基座模型在该场景中表现吃力。BIRD 将问题划分为简单、中等、挑战三个等级，SA-SQL 也同样在解决“挑战性难题”时表现出最显著的改进。同时，即使是简单和中等难度，SA-SQL 与基座模型相比也有一定提升，这主要源于大语言模型幻觉的减少，表明了预处理、后处理等阶段对 SQL 生成的普遍性价值。

3.4.3　消融实验

表 3.5　SA-SQL 框架中各组件的消融实验

流水线设置	*EX*	*Δ*
SA-SQL	66.6	-
w/o **预处理**	63.4	−3.2
w/o 模式链接	64.6	−2.0
w/o 值检索	65.2	−1.4
w/o 模式对齐	65.6	−1.0
w/o **推理**	-	-
w/o 少样本示例选择	64.2	−2.4
w/o 思维链推理策略	62.0	−4.6
w/o 模式对齐	65.6	−1.0
w/o **后处理**	63.0	−3.6
w/o 自纠正	65.4	−1.2
w/o 自一致性投票	64.2	−2.4

表 3.5详细概述了从方法中移除单个智能体和智能体组合模块时 BIRD 数据集上的执行准确率（EX）。本文专门研究了预处理、推理以及后处理阶段各个智能体的作用。此外，本文还研究了关键子模块，如模式对齐、少样本示例选择、关系代数推理策略和自一致性投票的影响，如表 3.5所示。从表中可以观察到：

- SQL 的执行准确率在去除任一智能体后均有下降，这表明不同智能体对 SQL 优化有明显的积极影响，证实了每个智能体的必要性。
- 消去预处理模块时，本文直接使用完整的数据库模式，没有进行模式链接

和值检索。可以观察到，模式链接和值检索对单个 SQL 的综合改进大致相当于预处理模块提供的改进，这表明它们的作用重叠较少。

- 在推理模块中，关系代数强化的思维链推理和少样本示例显著提高了单个 SQL 的结果，这表明提示中的信息量与大语言模型生成的 SQL 的有效性之间存在正相关关系。改进的思维链策略提供了 +4.6% 的提升，表现了其和 SQL 语法更高的契合度，可以激发大语言模型在该任务上的推理潜力。少样本示例学习同样有一定效果，本文推测少样本学习可以激发大语言模型的潜力，同时提高生成结果的稳定性。
- 消去后处理模块时，本文将直接使用推理后产生的 SQL 模板作为最终结果。可以观察到，自纠正和自一致性投票均对结果有一定增益。自纠正可以根据数据执行反馈作出相应调整，而自一致性投票有助于避免由低概率随机性引起的错误，可以提高大语言模型的性能上限。后处理为方案整体提供了 +3.6% 的提升，这证明了其在提高结果稳定性方面的关键作用。
- 模式对齐智能体在各个阶段都发挥了积极作用，显著提高了单个 SQL 的性能，从而展示了其独特的优化潜力。具体来说，在预处理阶段的对齐可以强化生成目标，减轻幻觉；在推理阶段的对齐可以校正 SQL 中细微的结构偏差，使 SQL 不仅可以执行还能执行正确。

表 3.6　提示模板信息消融实验

提示模板设置	EX	Δ
SA-SQL	66.6	-
w/o 列数据类型	66.4	−0.2
w/o 补充说明	65.5	−1.1
w/o 代表性值	63.4	−3.2
w/o 主键和外键	65.6	−3.0

同时本文还对 SQL 生成模板中除必要数据以外的添加信息进行了消融实验，如表 3.6所示，列数据类型对性能的影响较小，这可能是因为模型可以从列名和注释中推断出类型。由于 BIRD 基准测试的模式存在歧义，删除补充说明会显著影响其性能。此外，具有代表性的数据库值以及主键外键对模型整体的性能至关重要，前者有助于理解值的格式，后者则有助于准确生成 JOIN ON 子句。

3.4.4　少样本示例性能分析

本节研究推理少样本数 K_m 和自纠正少样本数 K_c 的最佳数量，将样本示例数从 0 到 5 依次递增，结果如图 3.3所示。为确保公平比较，当推理少样本数量动态变化时，自纠正少样本数量设置为 0，反之亦然。从图中可以看出，最终 SQL 的执行准确率随着推理少样本数量的增加而持续上升，相比之下，自纠正示例在候选数量为 1 时取得最佳结果，继续增加样本数量结果会有轻微下降。这可能是

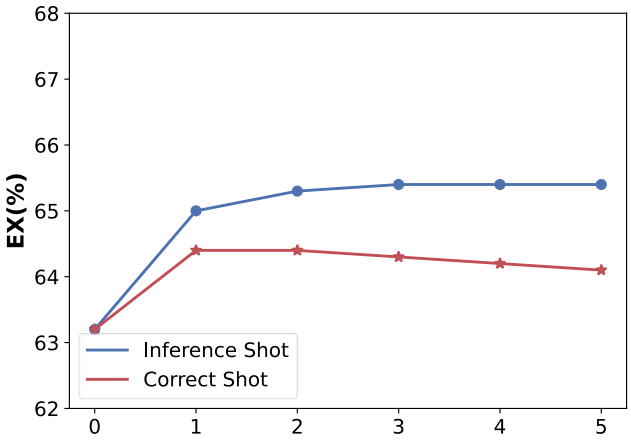

图 3.3　不同少样本提示数量的执行准确率比较

因为自纠正样本上下文过长，给模型带来了不必要的负担。综合考虑到性能表现和令牌开销，本文将 K_m 设置为 3，K_c 设置为 1。

3.4.5　思维链性能分析

本节对设计的思维链方法进行简要分析，比较为 SA-SQL 设计的关系代数思维链与传统的思维链方法（即“let’s think step by step”）以及无思维链情况下的性能。本文在实验中设置少样本示例数为 0，自一致性参与投票数为 1，以便直观比较思维链对 SQL 生成的影响。如表 3.7所示，使用传统思维链比不使用思维链在 SQL 生成方面有小幅改进，而加入关系代数的思维链比传统思维链有更大的提升。实验表明了关系代数在大语言模型推理过程中的约束作用，可以隐式地对齐自然语言表示的问题和 SQL 语法结构，减少结构错误对执行准确率造成的非必要影响。

表 3.7　BIRD 数据集上不同思维链策略执行准确率比较

方法	BIRD	Spider
不使用思维链	53.6	83.4
传统思维链	54.8(+1.2)	83.9(+0.5)
关系代数思维链	57.8(+4.2)	87.2(+3.8)

同时，为评估改进思维链的泛化性能，本文针对常用的大语言模型基座 ChatGPT、GPT4 和 Llama3-70B-Instruct 开展了实验。为直观比较性能，本文测试的两种思维链相较于基座模型仅在提示策略中有所不同，而不包含预处理和后处理模块。Spider 数据集涉及领域知识较少，更关注复杂推理产生的跨域 SQL 结构的正确性，故被用作测试数据集，结果如图 3.4所示。可以观察到，改进思维链的性能提升与模型的推理能力呈正相关。GPT4 表现出最强的能力，并且从推理策略的变化中获得了最高的收益。ChatGPT 和 Llama3-70B-Instruct 的性能提升也远远超过了基本的思维链（CoT）策略，不断逼近其推理能力的上限。实验结

果表明，改进思维链可以更加充分地激发大语言模型的推理潜力。

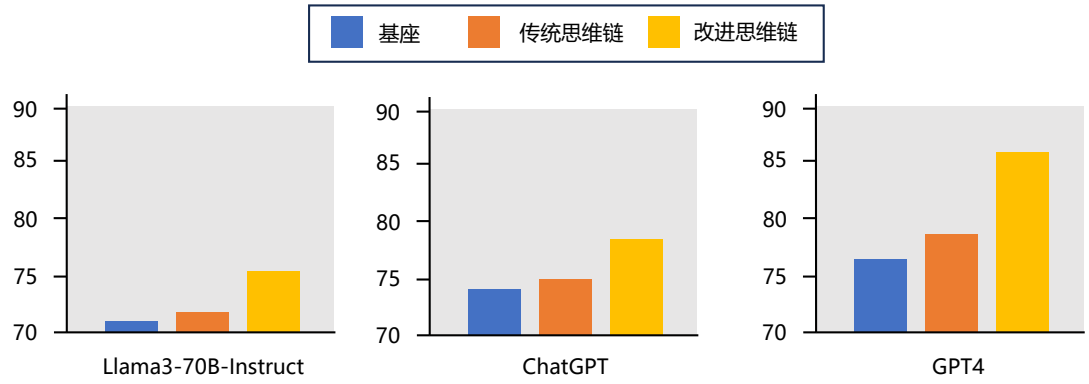

图 3.4　Spider 数据集上不同基座模型执行准确率比较

3.5　本章小结

本章介绍了 SA-SQL，这是一种使用模式对齐策略进行多智能体协作，从而提升 Text-to-SQL 任务性能的方法。为了强化大语言模型对提示的理解和遵循能力，本文提出了一个由三个主要阶段（预处理、推理和后处理）组成的标准 Text-to-SQL 框架，并引入了模式对齐策略来确保智能体之间的一致性。这种对齐策略有效地减少了智能体之间信息传递的损失，以及智能体生成结果中的幻觉。本文还开发了一种完善推理过程信息的思维链方法，将关系代数作为中间语言以提高推理过程和 SQL 生成的契合度，减少在复杂推理场景中传统思维链的错误累积。

本文在广泛使用的 BIRD 和 Spider 数据集上对 SA-SQL 进行了评估。结果表明，SA-SQL 在 BIRD 验证集上的执行准确率（EX）达到了 66.6%，超越了其他现有的提示工程方法。在 Spider 验证集上，SA-SQL 也取得了具有竞争力的结果，执行准确率达到了 90.8%。此外，本文还通过模块化消融实验验证了每个智能体的有效性，特别是模式对齐策略和改进的思维链推理智能体，它们对最终的性能提升做出了重要贡献。值得注意的是，SA-SQL 没有对大语言模型进行任何微调，而是直接在预训练的模型上实现，这使其具有强大的可移植性，可以泛化于各种专业领域的数据库查询任务，而无需进行额外的训练。

未来的工作可以从几个方面展开。首先，可以进一步优化提示工程，包括更精细地调整提示和更准确地检索列和值，从开放域中获取有助于 SQL 生成的知识以提高模型的性能。其次，可以探索更多的少样本方法和对齐策略，以进一步提高模型的稳定性和泛化能力。此外，结合有监督微调（SFT）等技术，可能会进一步提升 Text-to-SQL 任务的性能，这也是一个值得探索的方向。总之，SA-SQL 通过多智能体协作和模式对齐策略，显著提高了大语言模型在该任务上的性能。这项工作将为未来的 Text-to-SQL 研究和应用提供有价值的参考。

第 4 章　基于数据增强和多任务协作的 Text-to-SQL 算法

基于提示工程的方案虽然展现出了相对更优的性能，然而在许多资源受限场景（如参数规模受限、本地化部署、端到端延迟限制）中无法适用。为此，本章提出了基于微调这一技术路线的 Text-to-SQL 算法 EnhancedSQL。它的目的不再是追求性能的最优，而更关注在轻量模型上的性能相对表现和新领域的泛化性。具体地，本章 4.1 节介绍了当前微调方法存在的不足，4.2 节对提出的子任务做形式化定义，4.3 节中介绍了 EnhancedSQL 的数据构建、训练以及推理工作的具体细节，4.4 节中比较了 EnhancedSQL 和其他方案在轻量级模型上的性能比较，实验结果表明本章算法在多个场景中的泛化能力和相对较优的性能表现。

4.1　引言

大语言模型在 Text-to-SQL 任务被证明是一种有效的解决方案[11]。与早期工作[45,98-99]不同，基于大语言模型的方法[11-13]主要开发有效的提示工程技术，如上下文学习，以促使大语言模型理解数据库模式并生成准确的 SQL 查询。然而，在生成 SQL 时，准确对齐自然语言问题和数据库中的实体仍然具有挑战性，特别是在处理复杂的数据库模式或语义复杂的问题时[33]。为解决这些具有挑战性的场景问题，近期的研究开发了各种流程，以增强整个 SQL 生成过程并降低潜在的错误风险。这些改进包括模式链接[11]、自我校正[13]、思维链（CoT）[15]、可靠性投票[50]等技术。尽管这些方法取得了不错的成果，但它们通常依赖闭源模型，这在实际场景中部署大语言模型时可能会带来潜在的隐私风险，并产生高昂的成本。此外，虽然这些技术在 GPT-4 或其他大型大语言模型上表现良好，但在应用于较小的开源大语言模型时，其有效性可能会降低。这是由于较小的大语言模型理解复杂指令的能力有限，泛化性较低。

另一种方法是通过预训练或监督微调将通用大语言模型转化为专用大语言模型，注入与 Text-to-SQL 相关的知识[18-21]。然而，大多数基于训练的方法仅在监督微调阶段纳入训练集中给定的 SQL 生成任务，这使得模型能力高度依赖训练集，难以在广泛知识边界中具有鲁棒性。此外，在单一 SQL 生成任务上训练大语言模型，在理解指令方面存在性能下降的巨大风险，这可能会降低模型在 SQL 生成之外的其他重要 SQL 相关任务中的有效性。现有的数据集，如 Spider 等多为人工标注，不仅涉及到的领域多样性不足，在面对某些 SQL 特定结构时也难以避免标注人员的个人偏好，这对模式的泛化性带来潜在影响。考虑到现有提示或训练方法的局限性，受工作[100]启发，本文提出了一种用于 Text-to-SQL

生成的数据增强与多任务微调框架 EnhancedSQL。其动机如下：

- 增强训练数据的多样性，强化模型对 SQL 语法的理解，减少常见类型错误。
- 增强模型的 SQL 生成能力，同时保留其他相关任务如模式链接的能力。

生成高质量的 Text-to-SQL 微调数据应考虑两个条件。首先，训练集应具备多样性以促进跨领域泛化，同时使模型成功应用于新的领域或数据库。其次，训练数据应对齐人类偏好，减少系统性偏差，从执行反馈（尤其是错误）中学习。为实现这一目标，在数据准备阶段，本文探索了一种针对错误类型进行数据合成的方法，强化了大语言模型对于易错结构的修正能力；在训练阶段，本文探索了一种多任务监督微调范式，赋予大语言模型在 Text-to-SQL 任务中模式链接、SQL 生成、错误校正三个关键环节的能力；在推理阶段，本文采用了一种多任务协作提示方法，以选择性地、逐步地生成 SQL 查询，从而降低复杂 SQL 生成中出现幻觉的风险。本章的主要贡献总结如下：

- 在数据准备阶段引入数据增强策略，可以高效地合成不同领域数据。
- 在训练阶段构建多任务监督微调框架，赋予大语言模型多种与 SQL 相关的专业能力，尤其是针对错误的纠正能力。
- 在推理阶段设计多任务协作提示策略，通过将 Text-to-SQL 任务拆分为三个关键子任务，提升复杂问题的推理效果。

4.2　问题定义

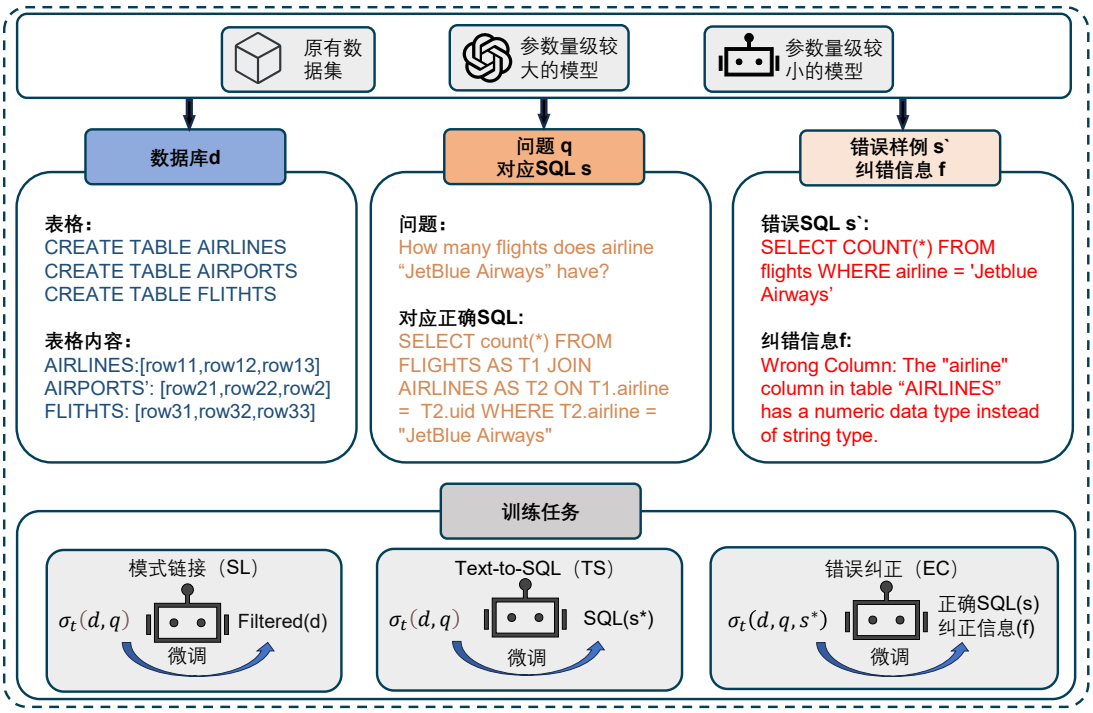

图 4.1　EnhancedSQL 中涉及的大语言模型和各个组件

在本文中，除了标准的 Text-to-SQL 任务，还在训练和推理阶段纳入了两个额外的与 SQL 相关的任务，包括模式链接、错误校正，如图 4.1所示。所有这些任务的定义如下：

- Text-to-SQL：旨在基于由数据库（d_i）和问题（q_i）构建的提示生成一个 SQL 查询（s_i^*），且 s_i^* 的执行结果与其真实的 SQL 查询（s_i）的执行结果一致。基于数据库和问题的提示格式化函数表示为 $\sigma_t(d_i, q_i)$。
- 模式链接（Schema Linking）：旨在为给定的问题（q_i）识别数据库（d_i）中的相关表和列，从而避免提示中出现冗长的信息，降低复杂度。研究[11]已证明这可以提高 Text-to-SQL 的性能。提示格式化函数表示为 $\sigma_s(d_i, q_i)$。
- 错误校正（Error Correction）：需要根据数据库 d_i，判断预测的 SQL 查询（s_i^*）的执行结果是否能正确回答问题（q_i）。如果不能，则要求大语言模型提供一个修正后的 SQL 查询（s_i）。提示格式化函数表示为 $\sigma_e(d_i, q_i, s_i^*)$。

从所提供的任务定义来看，模式链接和 Text-to-SQL 这两项任务有直接提升 SQL 查询质量的潜力，因为它们与 SQL 生成直接相关。此外，本文构造了错误纠正的任务，它对最终的 SQL 生成有间接贡献。错误纠正任务的动机源于大语言模型具备的反思和纠错能力。为了发挥这些任务的潜力，本文首先通过聚合多个任务进行有监督的微调，来增强大语言模型的专业能力。然后利用多个任务之间的协作，降低 SQL 生成过程中表结构链接错误或 SQL 子句错误等潜在风险，从而进一步提升性能。

4.3　算法框架

EnhancedSQL 算法主要涉及到数据准备、训练和推理三部分的工作，如图 4.2所示。它先依赖大型模型的领域知识合成多样化数据，再以轻量化模型作为过滤器从中提取典型错误。错误中的正负样例构成错误纠正数据集，帮助 Text-to-SQL 训练任务对齐偏好，同时利用大型模型判断其错误原因用于更进一步的错误纠正模型训练。

4.3.1　数据准备

1. 增强数据集构建

监督微调将显著提升模型生成恰当回复的性能，包括在 Text-to-SQL 任务中的表现。目前流行的跨域数据集（主要是 Spider 和 BIRD）需要人类专家进行标注，成本高昂。为降低成本并进一步扩大规模，本文借助其领域知识较为丰富的大型模型 GPT-4，通过提示来合成目标数据。鉴于跨域泛化是 Text-to-SQL 任务的核心挑战，本文设计了提示内容，促使 GPT-4 生成足够多样化的数据集，以此

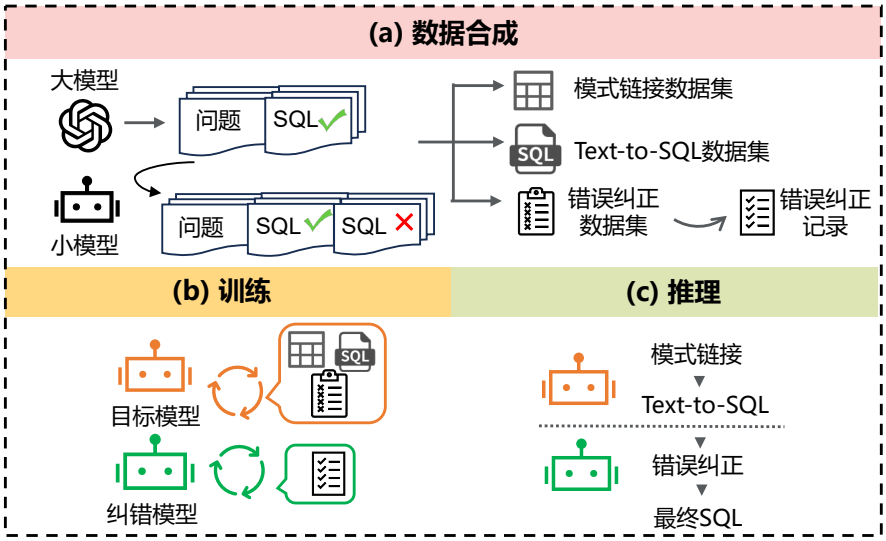

图 4.2　EnhancedSQL 算法框架图

鼓励数据多样性。指令首先要求领域多样性，明确排除出现频率过高的领域，从而引导 GPT-4 生成不仅多样，还能适应不同复杂程度的数据样例。接下来，受 DIN-SQL[11] 的对问题难度的划分启发，本文为生成问题定义了三种难度：

- 简单：聚焦单个表格的简单查询；
- 中等：涉及多个表格连接或聚合函数的复杂查询；
- 困难：需要深度理解的复杂查询，答案需运用多种高级特征

相较于 Spider 和 BIRD 数据集，新构建的数据集 $\bar{D}$ 中包含更多需要 JOIN 连接多表的操作和进行聚合函数计算的操作，其复杂度和深度相较于现有数据集更高。

2. **任务训练数据构建**

本文的多任务监督微调由一个主要任务（Text-to-SQL）和 4.2 节中定义的两个次要任务（即模式链接和错误纠正）组成，以赋予大语言模型专业能力。为此，本文需要从 $\bar{D}$ 为每个特定任务合成或构建监督微调数据集。

（1）对于 Text-to-SQL 任务的监督微调（SFT）数据，本文利用定义的函数 σ_t 来构建提示，真实的 SQL 查询作为响应。为 Text-to-SQL 构建的监督微调数据表示如下，其中 N_t 是相应的数据规模：

$$D_t = \{\sigma_t(d_i, q_i), s_i\}_{i=1}^{N_t} \tag{4.1}$$

(2) 对于模式链接任务的监督微调（SFT）数据，本文利用真实的 SQL 查询来提取对应数据库中的表和列作为响应，这一操作记为 $f(s_i, d_i)$，其中 f 是解析函数。如果存在 COUNT(*) 语句，本文仅保留对应表的主键。本文将该数据集表示为

$$D_s = \{\sigma_t(d_i, q_i), f(s_i, d_i)\}_{i=1}^{N_s} \tag{4.2}$$

（3）对于错误纠正任务的数据，本文需要生成典型的错误 SQL 查询，在 SQL

生成过程中直接对齐人类偏好。为寻找高质量错误样例，本文首先在大型模型（如 GPT-4）上测试增强数据集 $\bar{D}$ 的生成 SQL 是否与正确 SQL 相符，发现其生成的 SQL 语法结构规范，语义正确，难以生成有价值的错误数据。于是本文选择借助轻量化模型（如 LLama3-8B）的生成结果筛选数据集中错误的 SQL 样本，构建正负样例对运用直接偏好优化算法（Direct Preference Optimization，DPO）训练大语言模型。

为进一步提升纠正质量，本文还引入了专用的错误纠正模型。具体来说，本文为每个 {问题-SQL} 对构建一个正判别示例 $< \sigma_n(d_i, q_i, s_i), A_{pos} >$ 和一个负判别示例 $< \sigma_n(d_i, q_i, s_i'), A_{neg}(s_i) >$，其中 s_i 是真实的 SQL 查询，A_{pos} 和 A_{neg} 表示肯定和否定答案。A_{pos} 中包含模型对样本 SQL 的判别信息，而 A_{neg} 的内容除了判别信息之外，还包含对错误的分析归纳以及改正过后的 SQL 尝试。为获得用于构建 A_{neg} 的负判别示例，本文使用轻量化模型 Qwen2.5-7B 和 Llama3-8B 以零样本方式为数据集 $\bar{D}$ 中的问题生成 SQL 响应，然后将每个预测的 SQL 查询的执行结果与真实的 SQL 查询的执行结果进行比较。如果不匹配，则将预测的 SQL 查询视为负示例。接下来，本文使用 GPT-4 作为大型模型分析预测的 SQL 查询与真实 SQL 的区别，并归纳相应的错误原因 f。三者共同最终构建负示例的训练数据 A_{neg}，如图 4.1所示。此外，为丰富监督微调数据的多样性，本文还人为地在真实的 SQL 查询语句中引入了一些错误，如图 4.3所示。基于最近的一些研究探索，本文重点关注五种类型的错误：模式链接错误、嵌套错误、分组（GROUP BY）错误、连接（JOIN）错误以及符号错误，它们常常作为模型整体性的偏差批量出现，其具体信息如下:

- **模式链接错误**：指 SQL 查询语句中出现错误的表名和列名。本文通过随机制造拼写错误以及对表名或列名进行同义词替换来引入这类错误。
- **嵌套错误**：指在需要使用嵌套或集合操作（例如 UNION（并集）、INTERSECT（交集）和 EXCEPT（差集））时却未使用这些操作。本文通过随机删除这些关键字之前或之后的子 SQL 语句来破坏原有的 SQL 查询语句。
- **连接（JOIN）错误**：通常发生在使用 JOIN 操作的 SQL 查询语句中，表现为关注了错误的表名或列名。本文通过随机替换表名或列名来引入这类错误。
- **分组（GROUP BY）错误**：通常发生在使用 GROUP BY 操作的 SQL 查询语句中，表现为关注了错误的列名。本文通过随机替换 GROUP BY 之后的列名来引入这类错误。
- **符号错误**：一些小错误，例如错误的关键字、缺少逗号、缺少括号以及混淆函数名（如 COUNT（计数）、MAX（最大值）、MIN（最小值）等等）。

模式链接错误				
/* 问题 */ What is the post ID and the comments commented in the post titled by "Group differences on a five point Likert item"?	/* 问题提示 */ Title = 'Group differences on a five point Likert item';	/* 正确SQL */ SELECT T2.Id, T1.Text FROM comments AS T1 INNER JOIN posts AS T2 ON T1.PostId = T2.Id WHERE T2.Title = 'Group differences on a five point Likert item'	/* 错误SQL */ SELECT T2.`Id` AS CommentId, T2.`Text` AS CommentText FROM posts AS T1 INNER JOIN comments AS T2 ON T1.`Id` = T2.`PostId` WHERE …	/* 错误分析*/ Different table join order, use wrong table id
嵌套错误				
/* 问题 */ What is citizenship shared by singers born after 1955 and before 1945.	/* 问题提示 */	/* 正确SQL */ SELECT citizenship FROM singer WHERE born < 1945 INTERSECT SELECT citizenship FROM singer WHERE born > 1955	/* 错误SQL */ SELECT citizenship FROM singer WHERE born < 1945 and born > 1955	/* 错误分析*/ INTERSECT should have been used but was not
连接错误				
/* 问题 */ What is the total score of the posts edited by Yevgeny and include the user's website URL.	/* 问题提示 */ "Yevgeny" is the DisplayName; edited refers to LastEditorUserId	/* 正确SQL */ SELECT SUM(T1.Score), T2.WebsiteUrl FROM posts AS T1 INNER JOIN users AS T2 ON T1.OwnerUserId = T2.Id WHERE T2.DisplayName = 'Yevgeny' GROUP BY T2.WebsiteUrl	/* 错误SQL */ SELECT ... ON T1.`LastEditorUserId` = T2.`Id` WHERE T2.`DisplayName` = 'Yevgeny'	/* 错误分析*/ Use wrong db foreign key
分组错误				
/* 问题 */ What is the name of student who has the most pets.	/* 问题提示 */	/* 正确SQL */ SELECT T1.name FROM students as T1 INNER JOIN pets as T2 WHERE T1.id = T2.student_id GROUP BY T1.id	/* 错误SQL */ SELECT T1.name FROM students as T1 INNER JOIN pets as T2 WHERE T1.id = T2.student_id GROUP BY T1.name	/* 错误分析*/ Use wrong column in GROUP clause
符号错误				
/* 问题 */ What is the grade span offered in the school with the highest longitude?	/* 问题提示 */	/* 正确SQL */ SELECT GSoffered FROM schools ORDER BY ABS(longitude) DESC LIMIT 1	/* 错误SQL */ SELECT `GSoffered` FROM schools WHERE `Longitude` = (SELECT MAX(`Longitude`) FROM schools)	/* 错误分析*/ No information about ABS(longtitude)

图 4.3　五种常见错误类型样例

为了获取人工构造的负样本示例，本文会根据一定的概率选择某一种类型的错误，并将其引入到真实的 SQL 查询语句中。如果某个 SQL 查询语句不满足所引入的错误类型，比如没有分组（GROUP BY）操作，本文将随机选择其他类型的错误来继续构造负样本示例。最终的错误纠正专用数据集表示如下，其中 N_p 和 N_n 分别代表正负例子数量：

$$D_e = \{\sigma_n(d_i, q_i, s_i), A_{pos}\}_{i=1}^{N_p} \cup \{\sigma_n(d_i, q_i, s_i'), A_{neg}(s_i)\}_{i=1}^{N_n} \tag{4.3}$$

4.3.2　训练

现有的基于监督微调的方法主要关注 Text-to-SQL 任务，以提高最终性能。然而，先前基于提示的方法表明，与 SQL 相关的任务也可以有效地提高性能。遗憾的是，由于缺乏对这些相关任务的专门预训练或指令微调，开源大语言模型很难高精度地完成这些任务。本节将详细介绍在增强数据集上进行的多任务监督微调和直接偏好对齐，该数据集结合了两个广泛使用的跨域数据集以及根据上文中策略生成的数据。

本文首先采用标准的多任务监督微调过程对 Llama3-8B 模型进行微调，由主要任务 Text-to-SQL 和直接相关的模式链接任务组成，以赋予大语言模型专业能力。各项任务的专用数据集在 4.3.1 节中生成，将多任务监督微调数据集定义为 $D_M = D_t \cup D_s$，将生成的输入提示和目标响应分别表示为 x 和 y，T 表示为 y 的序列长度, 针对专用大语言模型的监督微调可以表示为最大化对数似然目标：

$$\mathbb{E}_{(x,y)\sim D_M}\left[\sum_{t=1}^{T}\log p_{\mathcal{M}}\left(y_t|y_{1:t-1},x\right)\right] \tag{4.4}$$

在有监督微调后，本文还需要针对错误纠正正负样例集进行直接偏好优化（DPO），以最大化模型在偏好样本上的优势。在完成上述训练任务后，可以得到一个专用的轻量化 Text-to-SQL 模型。除此之外，本文还引入了专用的错误纠正模型，它独立于训练目标模型，根据错误纠正的具体内容分析监督微调得到。为增强其客观性，纠正模型与训练目标模型使用不同的轻量化模型作为基座。

4.3.3　推理

为了充分利用大语言模型的这些专业能力，本文采用了一种多任务协作提示方法，以降低 SQL 推理过程中模式链接错误或 SQL 语法错误的潜在风险。本文的推理过程包含三个核心步骤，与 4.2 节中定义的任务相对应。

首先，为精简提示中的冗余信息，本文使用一种增强的模式链接策略。该策略利用大语言模型的模式链接能力，识别与解决用户问题相关的表和列，并借助模板 SQL 查询简化数据库。给定数据库 d_i 和用户问题 q_i，模板 SQL 是指利用完整模式信息预先生成的中间 SQL 查询，即 $\mathcal{M}(\sigma_t(d_i,q_i),d_i)$。最终简化后的数据库 $\tilde{d}_i$ 可表示为：

$$\overline{d}_i = \mathcal{M}\left(\sigma_s\left(d_i,q_i\right)\right) \uplus f\left(\mathcal{M}\left(\sigma_t\left(d_i,q_i\right),d_i\right)\right) \tag{4.5}$$

其中 f 是通过模糊匹配从 SQL 查询中提取表和列的解析函数，$\uplus$ 表示用于合并表和列的操作。这种合并操作有助于降低遗漏潜在相关实体（表或列）的可能性，并最大化与问题相关的信息，从而降低模式链接错误的潜在风险。模式链接提示模板 σ_s 如图 4.4 所示，Text-to-SQL 提示模板 σ_t 如图 4.5 所示。通过模式链接获取表和列后，本文整合精简的数据库相关信息 $\tilde{d}_i$ 和问题 q_i 进行 SQL 生成，得到中间 SQL 查询 s_i^*：

$$s_i^* = \mathcal{M}(\sigma_t(\tilde{d}_i,q_i)) \tag{4.6}$$

最终，为明确识别不正确的 SQL 查询，本文需要在专用的纠正模型上获取 SQL 纠正结果 A_{pos} 或者 A_{neg}：

$$A = \mathcal{M}_c(\sigma_e(d_i,q_i,s^*)) \tag{4.7}$$

如果回答表明 s_i^* 可以准确回答 q_i，s^* 保持不变；如 s_i^* 不能准确回答 q_i，本文会获取修正后的 SQL $\bar{s}_i$ 替换 s^*, 错误纠正提示模板 σ_e 如图 4.6 所示。最终本文使用 SQL 执行器验证结果，如果执行成功，则将其作为最终结果输出；若不成功，则将数据库报错信息 e 加入错误纠正的提示中，重新进行纠错任务。考虑到整个流程的端到端延迟，这一循环的最大迭代次数设置为 1。

```
Given the following database schema and question, your task is to extract the tables
and columns relevant to solving the question.

/* Examples */
…few-shot examples omitted…

/*Database schema*/
CREATE TABLE satscores (cds NUMBER, rtype TEXT, AvgScrMath Number…
PRIMARY KEY (cds));
CREATE TABLE schools (CDSCode NUMBER, City TEXT, Virtual TEXT…
PRIMARY KEY (CDSCode));

/* Sample rows of table */
satscores: [(110011, 'D', 418), (110012, 'S', 546), (110013, 'S', 387)]
schools: [(109835, 'Hayward', 'P'), (112607, 'Oakland', 'N'), (118489, 'Berkeley', 'F')]

/* Question */
How many schools with an average score in Math greater than 400 in the SAT test are
exclusively virtual?

/* Question hint */（仅在BIRD数据集中使用）
Exclusively virtual refers to Virtual = 'P'

Output:
```

图 4.4　模式链接任务的提示模板

```
Given the following database schema and question, your task is to write a valid SQL
query whose execution results can accurately answer the question.
Note: Answer the question by a SQL query only with no explanation.

/*Database schema*/
CREATE TABLE satscores (cds NUMBER, rtype TEXT, AvgScrMath Number…
PRIMARY KEY (cds));
CREATE TABLE schools (CDSCode NUMBER, City TEXT, Virtual TEXT…
PRIMARY KEY (CDSCode));

/* Sample rows of table */
satscores: [(110011, 'D', 418), (110012, 'S', 546), (110013, 'S', 387)]
schools: [(109835, 'Hayward', 'P'), (112607, 'Oakland', 'N'), (118489, 'Berkeley', 'F')]

/* Question */
How many schools with an average score in Math greater than 400 in the SAT test are
exclusively virtual?

/* Question hint */（仅在BIRD数据集中使用）
Exclusively virtual refers to Virtual = 'P'

Output:
```

图 4.5　Text-to-SQL 任务的提示模板

```
Your task is to determine whether the SQL query can answer the given question according to
the following database schema. If the execution results cannot correctly answer the question,
please analyze reason for the error and give me the correct SQL query.

/* Examples */
…few-shot examples omitted…

/*Database schema*/
CREATE TABLE satscores (cds NUMBER, rtype TEXT, AvgScrMath Number…
PRIMARY KEY (cds));
CREATE TABLE schools (CDSCode NUMBER, City TEXT, Virtual TEXT…
PRIMARY KEY (CDSCode));
/* Sample rows of table */
satscores: [(110011, 'D', 418), (110012, 'S', 546), (110013, 'S', 387)]
schools: [(109835, 'Hayward', 'P'), (112607, 'Oakland', 'N'), (118489, 'Berkeley', 'F')]
/* Question */
How many schools with an average score in Math greater than 400 in the SAT test are
exclusively virtual?
/* Question hint */（仅在BIRD数据集中使用）
Exclusively virtual refers to Virtual = 'P'

/* SQL query */
SELECT COUNT(DISTINCT T2.School) FROM satscores AS T1 INNER JOIN schools AS
T2 ON T1.cds = T2.CDSCode WHERE T2.Virtual = 'P' AND T1.AvgScrMath > 400

/* Execution exception */
...information omitted...
Output:
```

图 4.6　错误纠正任务的提示模板

4.4　实验结果分析

4.4.1　实验设置

1. 基准测试

为评估本文的方法，本文在五个基准测试上进行了广泛实验，以验证其有效性。这些基准测试包括两个广泛使用的跨域基准测试 Spider[8] 和 BIRD[33]，以及三个从 Spider 派生的测试集 Spider-SYN[101]、Spider-DK[7] 和 Spider-Realistic[102]。Spider 的训练集包含 7000 个 {问题-SQL} 对，验证集包含 1034 个对，测试集包含 2147 个对，涵盖了近 200 个数据库和 138 个领域。BIRD 是最近提出的一个基准测试，其训练集、验证集和测试集分别包含 9428 个、1534 个和 1789 个对。与 Spider 相比，BIRD 包含更复杂的数据库、更具挑战性的问题和外部知识，使其更具挑战性。对于派生的变体，Spider-SYN 将 Spider 验证集问题里的一些关键词替换为同义词，Spider-DK 引入了一些领域知识推理挑战，而 Spider-Realistic 则删除了 Spider 验证集中对列名的明确提及。这些变体都模拟了现实世界的场景，以便对本文训练后的模型进行更全面的评估。

2. 评估指标

遵循先前的研究[18-19]，本文使用执行准确率（EX）和测试套件准确率（TS）来评估 Text-to-SQL 模型在 Spider 及其变体基准测试上的性能。对于 BIRD 基准测试，按照其官方设置，本文报告 EX 和一个名为有效效率得分（VES）的指标，该指标考虑了执行效率，以此来评估性能。

3. 实现细节

本文选择了流行的大语言模型 Llama3-8B-Instruct[103]作为训练基座。为确保可重复性，本文使用 Llama-Factory[104] 框架进行监督微调（SFT）和直接偏好优化（DPO）。本文在配备 A100 GPU 的设备上进行实验，批量大小设置为 64。大语言模型使用 AdamW 优化器进行两个训练轮次的微调，初始学习率设为 1e-5，并通过余弦退火调度器在训练结束时衰减至 0。在推理过程中，温度设置为 0 以确保结果的可重复性。

4. 对比基线

本文对比基线可分为三组，即使用 GPT-4 的提示方法、使用开源大语言模型的提示方法，以及使用开源大语言模型的基于微调的方法。第一组包括 DIN-SQL[11]、MACSQL[13]、DAIL-SQL[12]、MCS-SQL[14]，以及相应的闭源大语言模型 GPT-4[74]。对于使用开源大语言模型的基线，本文选择了一些流行的大语言模型并报告其零样本性能。此外，本文还将 DIN-SQL 和 MAC-SQL 等提示方法应用于 Llama3[103]以探索其稳健性。最后一组以专门的大语言模型为代表，包括 DTS-SQL[11]、CODES[18]、SENSE[19]，以及在 Spider 和 BIRD 训练集上微调的基座大语言模型。

4.4.2　结果分析

1. 主要实验结果

在表 4.1中，本文展示了 EnhencedSQL 和各类基线模型在 Spider 和 BIRD 验证集上的性能表现，其中最优方案加粗显示，次优方案加下划线表示。从这些结果中可以看出，基于提示的基线模型在 GPT-4 的帮助下取得了不错的效果，但在轻量化模型中，其有效性明显受限。相比之下，本文的训练方法有效地提升了轻量化模型的性能。在 Spider 验证集上，EnhancedSQL 为 Llama3-8B 带来了 3.8% 的绝对性能提升。同时，EnhancedSQL 在大多数指标上，取得了基于微调开源大语言模型的方法中最佳结果。特别是在 BIRD 数据集上，本文的方法（部署于 8B 模型）以 57.6% 的执行准确率（EX）超过了一些现有的基于提示的方法（如 DIN-SQL、DAIL-SQL），极大地缩小了与使用 GPT-4 的现有方法之间的差距。

表 4.1　Spider 和 BIRD 数据集上不同方法性能比较

方法	Spider		BIRD	
	Dev-EX	Dev-TS	Dev-EX	Dev-VES
基于 *GPT-4* 的提示工程				
GPT-4[74]	72.9	64.9	46.4	49.8
DIN-SQL + GPT-4[11]	82.8	74.2	50.7	58.8
DAIL-SQL + GPT-4[12]	83.5	76.2	54.8	56.1
MAC-SQL + GPT-4[13]	86.8	78.8	59.4	66.2
MCS-SQL + GPT-4[14]	89.5	81.0	63.4	64.8
基于开源大语言模型的提示工程				
Mistral-7b[105]	56.8	47.3	22.5	27.8
Llama3-8B[103]	69.3	58.4	32.1	31.6
DIN-SQL + Llama3-8B	48.7	39.3	20.4	24.6
MAC-SQL + Llama3-8B	64.3	52.8	40.7	40.8
基于微调开源大语言模型				
Llama3-8B + SFT[103]	82.4	76.2	53.1	59.0
DTS-SQL-7B[79]	82.7	78.4	55.8	60.3
CODES-7B + SFT[18]	85.4	80.3	57.2	58.8
SENSE-7B[19]	83.2	**81.7**	51.8	54.2
EnhancedSQL + Llama3-8B	**86.2**	81.0	**57.6**	**60.4**

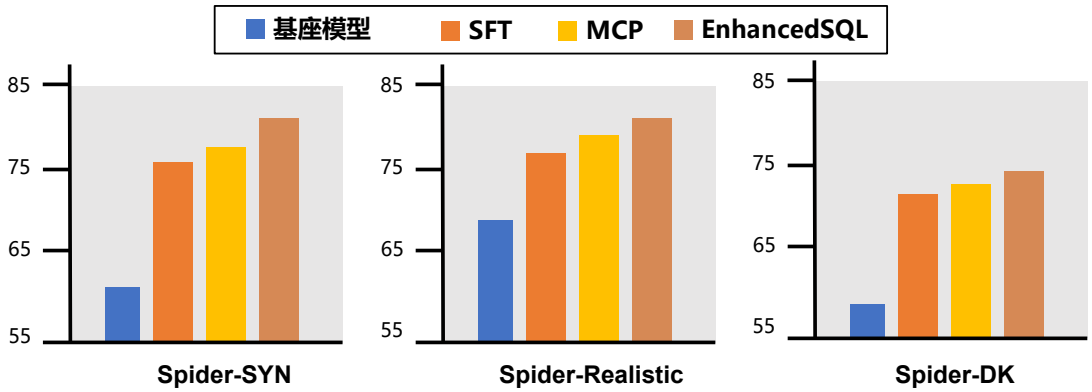

图 4.7　Spider 变体数据集不同训练方法性能表现

2. Spider 变体数据集上的结果

图 4.7展示了在源自 Spider 的基准测试上的性能表现，选取 Llama3-8B-Instruct 作为基座模型，本文分别采用 3 种不同的训练方法：

- SFT：仅使用对应数据集的 Text-to-SQL 训练集进行有监督微调，单步推理；
- MCP：在微调基础上加入模式链接和错误纠正任务，多任务协作逐步推理；
- EnhancedSQL：本文提出方法，多任务监督微调后进行多任务协作推理

实验结果表明，在所有变体数据集中 EnhancedSQL 均取得了训练方法中最优的表现，证明了模型泛化到全新数据集时通过数据增强学习和纠错的必要性。相较于单步推理，借助多任务协作提示明确子任务目的，可以使最终结果更加稳定，进而提升性能。此外，多任务协作提示对经过单一 Text-to-SQL 监督微调的

模型仍能带来性能提升，这证明了进行多任务协作的必要性。

4.4.3　消融实验

在本节中，本文首先对所有基准测试进行全面的消融研究，以探究每个关键组件的影响，验证本文方法的有效性。此外，本文还进行了深入分析，以探索 EnhancedSQL 的可转移性和上限性能。除非另有说明，所有结果均在 Llama3-8B-Instruct 模型上得出，并使用执行准确率（EX）分数在 Spider 和 BIRD 验证集上进行评估，训练阶段 MSFT 指代多任务监督微调，MCP 指代多任务协作提示策略；推理阶段 SL、TS、EC 分别指代模式链接、Text-to-SQL 和错误纠正三个子任务。

1. 数据准备阶段

表 4.2　数据来源的消融实验结果

数据	Spider	BIRD
EnhancedSQL	86.2	57.6
Spider+BIRD	80.4	52.8
Spider	77.5	38.3
BIRD	69.2	51.6

EnhancedSQL 使用合成的数据集来扩展数据的多样性，定位系统化偏差错误。表 4.2 测试了使用不同的训练集对模型最终结果的影响。可以看出，使用单一的训练集难以泛化至其他不同风格的域中，而 EnhancedSQL 填补了人工合成数据集中留下的空白，具有更强的泛化性。

2. 训练阶段

表 4.3　训练方法的消融实验结果

方法	Spider	BIRD
EnhancedSQL	86.2	57.6
训练后模型		
w/o MCP（仅在目标模型中进行单步推理）	83.6	53.8
w/o MSFT（传统单一任务微调但仍使用 MCP）	83.8	56.3
w/o MSFT+MCP（仅进行传统单一任务微调和单步推理）	83.4	52.9
基座模型		
w/o MSFT（仅使用多任务协作提示）	75.0	42.7
w/o MSFT+MCP（仅使用基座模型能力）	69.3	32.1

表 4.3 报告了训练过程中各种变体方案的消融实验结果，主要可分为训练后模型和基座模型两类。训练后模型尝试去除训练过程中除 Text-to-SQL 以外的任务，而基座模型中报告了传统的单步推理和多任务协作推理作为性能比较。从结果中可以发现，多任务协作提示对于训练后模型和基座模型的准确率均有较为显著提升，这表明在处理困难问题时，轻量化模型更适合完成拆分的具体任务再组合；同时，多任务监督微调也对最终模型的准确率有一定积极影响。

3. 推理阶段

表 4.4　推理任务的消融实验结果

方法	Spider	BIRD
训练后模型	86.2	57.6
w/o SL	83.9	54.8
w/o EC	85.8	56.1
w/o SL+EC	83.6	53.6
基座模型	75.0	42.7
w/o SL	72.3	38.2
w/o EC	73.4	36.8
w/o SL+EC	69.3	32.1

EnhancedSQL 在推理过程中使用三个子任务进行多任务协作提示，能够激发并充分发挥多任务协作在准确生成 SQL 方面的优势。为此，本文对多任务协作提示的各个环节进行消融实验，以探究每个任务的影响。消融实验同样分为经过多任务监督微调（MSFT）后的大语言模型和基座模型两类，结果如表 4.4所示。从实验结果中可以看到，错误纠正（EC）对两类模型均有一定的性能提升，这来源于它对模型固有偏差的修复。同时模式链接（SL）在性能提升方面起着主导作用，但如果没有完整的多任务协作提示流程，它仍无法实现最佳性能，这体现了多个任务之间的互补性。

4.5　本章小结

本章提出了一种基于微调的数据增强与多任务协作算法，以激发轻量化模型在 Text-to-SQL 任务中的潜力，缩小与基于 GPT-4 等大型模型的现有解决方案之间的差距。本文首先针对数据集和错误样例数量不足的问题，应用了一种数据增强方法构建了新领域和大类错误上的样例；接下来通过显式学习多个与 SQL 相关的任务并进行多任务协作，拆解 SQL 生成过程，依靠训练得到的错误案例进行自我纠错。本文将该方法应用于当下多个开源大语言模型，在 Text-to-SQL 基准测试中证明了其有效性和优越性，在多个领域的任务中实现了可观的执行准确率。本文未来计划探索更多与 SQL 相关的任务，更大规模的大语言模型，以及更高效的协作框架，以实现更稳健的 Text-to-SQL 解析任务。

第 5 章　基于 SQL 表分解的表格问答算法

Text-to-SQL 虽然可以解决基于关系型数据库的问答任务，但在更大的表格域中无法适用，这为表格问答的性能提升带来了挑战。为此，本章提出了基于 SQL 表分解的表格问答算法 DesTab。它可以在灵活的文本推理与简练的程序辅助推理（SQL）中动态选择，提升大语言模型针对大型自由格式表格的性能表现。具体地，本章 5.1 节介绍了当前表格问答任务面对的挑战，5.2 节对提出的子任务做形式化定义，5.3 节介绍算法框架和实现细节，5.4 节对 DesTab 和其他方案在大型自由格式表格数据集进行了性能比较，展现了 Text-to-SQL 任务对通用表格问答任务的性能提升。

5.1　引言

常见的表格理解任务包括表格问答[22-23]、表格事实验证[24-25]、表格到文本生成[26-27]。在这些任务中，表格问答是最具挑战性的任务之一，它要求模型同时理解自然语言问题和相应的自由格式表格数据。相较于数据库中的结构化表格，自由格式表格，如网页表格[106]包含的范围更广，缺乏预定义的模式，这使得前文中基于结构化查询（SQL 生成）的方法无法适用。

随着大语言模型展现出强大的性能，大量研究致力于自由格式表格问答[28,86,90]。然而，随着数据需求的持续增长[106-107]和表格规模的迅速扩大，处理大型自由格式表格所需的时间和计算资源显著增加。更长的上下文还加剧了大语言模型产生幻觉的风险[108]，在任务复杂时这种幻觉更为严重。提升大语言模型在大规模高复杂度数据场景下的表格推理能力已成为当前研究的重点。

近期研究主要通过分解策略应对大型表格问答挑战：第一种方法仍以自然语言引导大语言模型进行文本推理[87,90,109]，将复杂任务分解为多个简单子任务，基于子任务输出推断最终答案。但现有方法在处理复杂问题时逻辑推理能力有限，且仍需要预先输入完整表格，在大型表格场景下效率低下。如图 5.1所示，在将全部表格信息作为上下文输入后，大语言模型独立理解子任务间逻辑关系的难度进一步提升，已经无法准确地去除噪声推理正确答案。此外，子任务的分解与解答均需大语言模型持续参与，这意味着在处理大型表格时将产生高昂的时间和计算成本。第二种方法利用程序辅助推理[64,110]，通过生成 SQL 或 Python 代码提取相关子表，以支持最终答案的推理。然而，这种方法存在准确性低和可靠性差的问题。如图 5.1 所示，生成的 SQL 错误地将“cities in Tel Aviv”映射为 `District='Tel Aviv'`，而 `District` 字段实际包含更长的描述性语句，

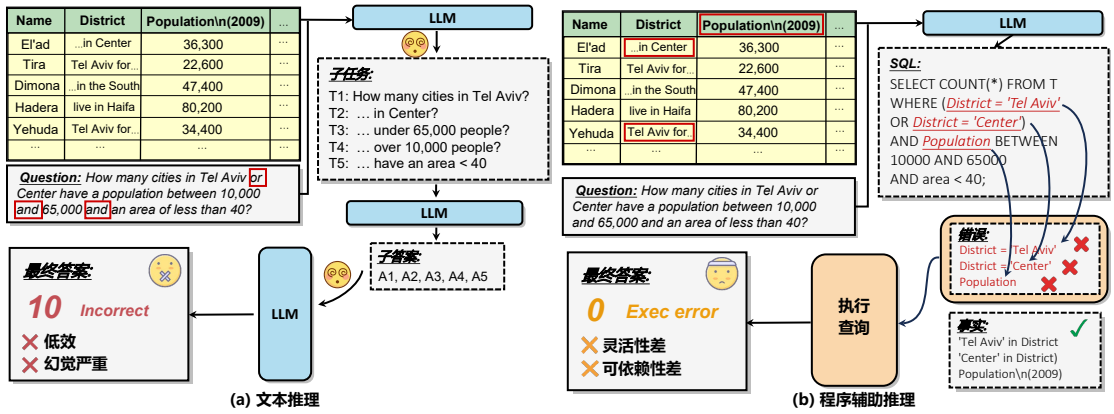

图 5.1　不同表格问答策略的样例学习

正确的条件应为 `Tel Aviv in District`。此外，`Population<65000` 这一子句也会查询失败，因为表中的字段名为 `Population\n(2009)`，导致查询无效。这些字段映射和条件解释方面的问题，显著影响了程序辅助推理在大规模表格中的可靠性。总体而言，现有方法面临两个核心局限：（1）文本推理效率低下，易受噪声干扰；（2）程序辅助推理不灵活、不可靠。

为应对这些挑战，本文展开了大语言模型推理性能与表格规模关系的研究，结果表明无论使用何种模型，其整体性能往往会随着表大小的增加而下降。对于较弱的大语言模型，性能下降更为明显。同时本文针对表格问答中的错误样例，将原有表格替换为显式构建的仅包含问题所需相关信息的子表格，结果表明此时表格的理解变得明显更容易，这与之前的发现和直觉一致。程序辅助推理，如 Text-to-SQL 方法虽然语义不够灵活，无法直接解决自由格式表格问答问题[111]，但生成的 SQL 查询规范性强，可以被视作表格分解的计划方案，有助于提取仅包含回答问题所需信息的相关子表格。

基于这些观察，本文提出了基于 SQL 分解的表格问答算法 DesTab，用于优化大型表格和复杂问题的推理。该方法首先对原始表格进行清洗和聚焦，以确保数据完整性和任务相关性。这一过程中提取的关键信息将作为大语言模型可学习的表格内容示例，引导 **SQL 生成器**（基于大语言模型）生成 SQL 查询。接下来，**表格分解器**（基于规则的解析器）并不会直接执行这些查询来检索答案，而是将它视为表格分解路径规划，从中提取关键信息：列名、值和条件，以形成操作序列指导表格分解，最终生成一个仅包含回答问题所需数据的子表格。最后，子表格被传递给**答案生成器**以生成最终答案。由于 SQL 生成器生成的 SQL 查询可能不准确，本文引入了 **SQL 验证器**，这是一个额外的基于大语言模型的模块，它执行自我验证和迭代优化以提高 SQL 质量，并进一步提高表格分解的准确性。综上所述，本文做出了以下贡献：

- 本文提出了 DesTab，这是一个旨在提高大型自由格式表格问答性能的模

型。它通过利用 SQL 生成器生成的 SQL 查询，框架中的表格分解器和验证器协同完成提取无噪声的相关子表格，显著提高了最终答案生成的准确性。

- 本文在两个通用数据集和两个大型表格数据集上进行了全面评估，实验结果表明，DesTab 在所有数据集中均取得了显著改进，性能优于目前所有竞争方案。这表明了 Text-to-SQL 任务虽然无法直接适用于大型自由格式表格推理，但仍能通过合理设计为通用表格问答，尤其是大型表格问答做出贡献。

5.2　问题定义

给定表格 T 和关于 T 中数据的问题 Q，表格问答任务的目标是开发一个能够为 Q 生成准确答案的模型。问题 Q 以自然语言表示，表格 T 也用自然语言表示为一个标记序列，其中各个单元格用特殊字符（如‘|’）分隔，行用换行符分隔。模型的性能通过将其生成的答案与真实答案进行基于标记的比较来评估。为了有效地处理大型表格上的表格问答，本文解决了两个关键子问题：Text-to-SQL 和表格分解。

5.2.1　Text-to-SQL

这个问题涉及从问题 Q 和与表格 T 对应的表头信息 $H = (h_1, h_2, ..., h_{|H|})$ 生成一个 SQL 查询 S，其中每个 $h_i \in H$（$i \in [1, |H|]$）表示 T 中的一列。目标是生成一个基于 H 准确捕捉 Q 意图的 S。不同于三、四两章中的 Text-to-SQL 任务，本章重点不是执行 S，而是解析 S 以提取用于表格分解的关键组件，包括列名 C、条件 P（例如 `LIKE` 子句）和 S 的过滤操作中涉及的值 V。

5.2.2　表格分解

这个问题旨在根据提取的组件：列名 c、条件 P 和值 V，从表格 T 中提取一个子表格 T_{sub}。给定表格 T，其中 $C = \{c_1, c_2, ..., c_{|C|}\}$ 表示选定列的集合，$P = \{p_1, p_2, ..., p_{|P|}\}$ 表示过滤条件，$V = \{v_1, v_2, ..., v_{|V|}\}$ 是条件中使用的值，目标是构建 T_{sub}，使得它仅保留 c 中指定的列，并仅包含满足 P 的行。这个过程使用 Python DataFrame 操作来实现，它可以处理列选择和行过滤。

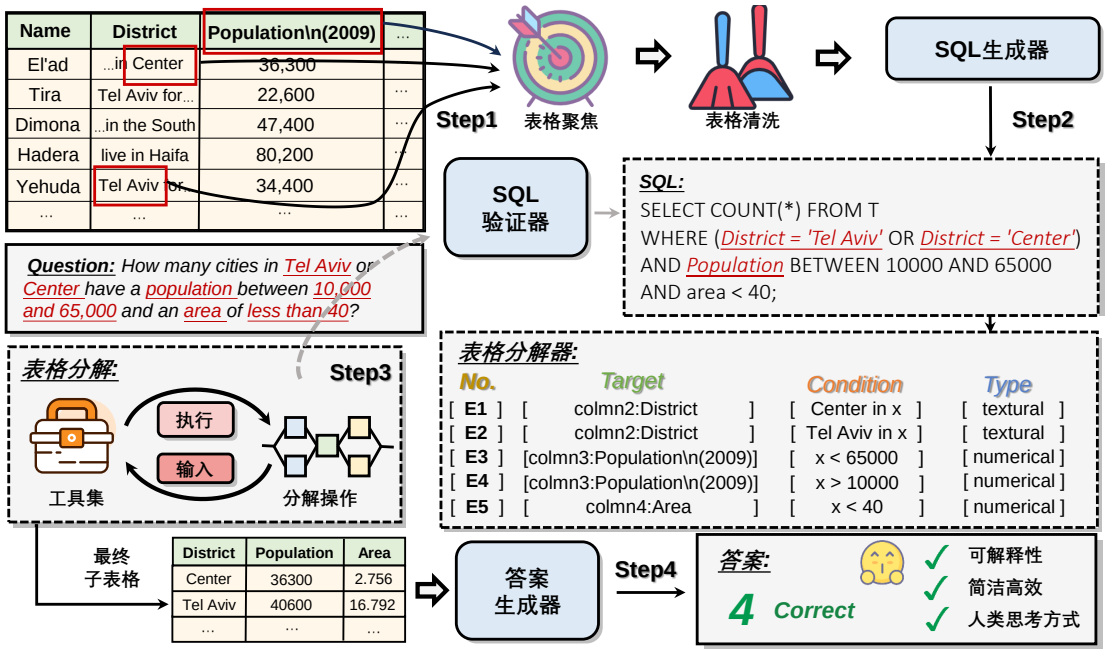

图 5.2　DesTab 整体框架图

5.3　算法框架

DesTab 分四个阶段处理表格问答：表格清洗、SQL 生成、表格分解、答案生成，如图 5.2所示。首先在表格清洗阶段（5.3.1 节）要规范表的结构，根据问题筛选相关行和列以构建后续 SQL 生成的示例项；SQL 生成阶段（5.3.2 节）涉及到 SQL 生成器和 SQL 验证器。SQL 生成器根据问题 Q、表格 T 的表头信息和示例项生成一个原始 SQL 查询 S_{raw}。然后，SQL 验证器对 S_{raw} 进行验证和优化，生成最终的 SQL 查询 S。表格分解阶段（5.3.3 节）使用表格分解器来解析 S，并提取三元组 (C, P, V) 序列，分别表示列名、条件和值。这个三元组将转化为 Python DataFrame 操作，以构建一个仅包含与问题相关的行和列的子表格 T_{sub}。操作表格中如果出现错误则将相关信息加入提示，并经过 SQL 验证器重新生成；答案生成阶段（5.3.4 节）使用答案生成器，根据问题 Q 和子表格 T_{sub}，利用大语言模型生成最终答案 A。

5.3.1　表格清洗

在表格问答任务中，与任务相关的信息通常稀疏地分布在大量无关内容中，这使得大语言模型难以准确聚焦关键数据，尤其是在包含众多字段的大型表格中[108,112]。为解决这一问题，现有方法通常采用截断策略[110]或基于嵌入的匹配方法[113]来提高任务相关数据的密度。然而，这些方法往往不可避免地导致部分关键信息的丢失。为此，本文对原始表格进行清洗和聚焦处理，以确保数据完整性和任务相关性。首先对 T_{raw} 执行表格清洗，标准化数值和日期字段的格式。例如，将“60,160”等数值规范为“60160”，并将“26 January 2003”和“January

26, 2003”等日期表达式转换为统一格式“2003-01-26”，由此得到表格 T。随后，本文对 T 进行表格聚焦处理：参照 Chase-SQL[114] 方法，通过提示大型语言模型从问题中提取一组关键词。对于每个关键词，本文使用局部敏感哈希（LSH）[115] 检索初始候选数据，并基于嵌入相似度和编辑距离对候选数据进行重排序，选取排名前 K 的结果构建是示例用子表 T_i（K 为可调超参数）。T 确保了数据完整性，使验证阶段更可靠稳定；而 T_i 则确保了任务相关性和高效性，使 SQL 生成能够更有效地聚焦目标信息。

5.3.2　SQL 生成

本文首先使用 SQL 生成器生成一个原始 SQL 查询 S_{raw}，并将其传递给 SQL 验证器进行验证。SQL 生成器使用基于骨干大语言模型的少样本提示，包括了表格 T 的表头信息、根据相关性排序得到的表格样例行 T_i 和问题 Q，如图 5.3。然后，这个提示用于查询大语言模型，生成原始 SQL 查询 S_{raw}。如果 S_{raw} 验证失败，SQL 验证器会对其进行优化，生成最终的 SQL 查询 S。

```
Given the following database schema and question, your task is to write a valid SQL
query whose execution results can accurately answer the question.
Note:
-    Do not add any explanation after the SQL.

/* Examples */
…few-shot examples omitted…

/* Table columns*/
[TABLE_COLUMNS]

/* Sample rows of table */
[SAMPLE_ROW_VALUE]

/* Question */
[Question]

Corresponding SQL:
```

图 5.3　SQL 生成器提示模板

由于表格分解高度依赖于 SQL 查询 S_{raw} 的质量，因此在整个框架中引入 SQL 验证器来验证 S_{raw} 是否能得到正确的子表格。验证过程如下：

1. 初始子表格提取：使用表格分解器（5.3.3 节）解析 S_{raw}，得到原始子表格 T_{sub}。
2. 验证提示：使用 T_{sub}、表格 T 的表头信息和问题 Q 提示 SQL 验证器，它会判断 T_{sub} 是否包含回答问题 Q 所需的所有信息，如图 5.4。
3. 决策：如果 SQL 验证器输出 `[True]`，则接受 S_{raw} 作为最终的 SQL 查询 S；否则，SQL 验证器通过提示大语言模型生成改进的 SQL 查询来优化 S_{raw}，如图 5.5。在重新生成 SQL 时示例行应重新选择以扩大大语言模型

接收的表格信息范围。

4. 迭代控制：这个验证和优化过程最多可以进行 n 轮，默认 n = 1 以提高成本效率。

在这个 SQL 验证阶段结束时，本文得到了最终的 SQL 查询语句 S，它将作为表格分解的全局指导有序执行表格操作序列。

```
Given the following database schema and sub-table, your task is to determine whether
the sub-table is sufficient to answer the given question.
Note:
- Complete table contains the following columns: [TABLE_COLUMNS]
- Give me the answer in format "Final Answer: True / False" form (should be either
  True or False, without any explanation)

/* Sub-table columns*/
[SUB_TABLE_COLUMNS]

/* Sub-table content*/
[SUB_TABLE_ROW_VALUE]

/* Question */
[QUESTION]

Final Answer:
```

图 5.4　SQL 验证器提示模板

```
Given the the question and the table schema, your task is to refine the given SQL.
Note:
- The current SQL query cannot solve the problem well. Please optimize it and return the
  optimized SQL directly.

/* SQL*/
[SQL]

/* Table columns*/
[TABLE_COLUMNS]

/* Sample rows of table */
[RESELECT_SAMPLE_ROW_VALUE]

/* Question */
[QUESTION]

Correct SQL:
```

图 5.5　SQL 重新生成提示模板

5.3.3　表格分解

表格分解器将表格 T 和在 SQL 生成阶段获得的 SQL 查询 S 作为输入，并采用基于规则的方法提取子表格 T_{sub}。首先需要从 S 中提取关键信息，本文采用基于正则表达式的规则驱动方法。使用正则表达式解析 S，提取列名 C、条件 P（例如过滤子句）和值 V（例如过滤子句中的值）。接下来利用提取的三元组 (C, P, V) 进行表格分解。先将表格 T 转换为 Python DataFrame，然后根据 (C, P, V) 应用 DataFrame 操作提取行和列，得到最终的子表格 T_{sub}。Python DataFrame 操作与

SQL 查询之间的对应关系如表 5.1所示。

表 5.1　Python DataFrame 和 SQL 之间对应关系

SQL 类型	SQL 表示	DataFrame 示例
基本查询语句	GROUP BY	df.groupby('column')
	HAVING	df.groupby().filter(lambda x: len(x) > x)
	ORDER BY	df.sort_values('age', ascending=True/False)
	LIMIT	df.head(10)
逻辑运算	AND	df[(df['age'] > 25) & (df['salary'] > 500)]
	OR	df[(df['age'] > 25) | (df['salary'] > 500)]
	NOT	df[~(df['age'] > 25]
聚合计算	SUM	df['salary'].sum()
	AVG	df['salary'].mean()
	COUNT	df['salary'].count()
	MIN	df['salary'].min()
	NAX	df['salary'].max()
比较	=	df[df['age'] == 30]
	!=	df[df['age'] != 30]
	>	df[df['salary'] > 3000]
	<	df[df['salary'] < 5000]
	>=	df[df['salary'] >= 3000]
	<=	df[df['salary'] <= 5000]
	BETWEEN AND	df[(df['salary'] >= 3000) & (df['salary'] <= 5000)]
	IN	df[df['country'].isin(['chinese', 'english'])]
	NOT IN	df[~df['country'].isin(['chinese', 'english'])]
	IS NULL	df[df['chinese'].isna()]
	IS NOT NULL	df[df['chinese'].notna()]
	LIKE	df[df['name'].str.contains('A')]

5.3.4　答案生成

在此阶段，利用基于大语言模型的强大语义解析和推理能力来执行端到端的表格问答。构建一个如图 5.6的提示，包含 T_{sub} 和 Q，将其输入到答案生成器（基于大语言模型）中，生成最终答案 A。

```
Given the the table, your task is to answer the question.
Note:
- Give me the answer in format "Final Answer: AnswerName1, AnswerName2..." form (should
  be a number or entity names, as short as possible, without any explanation)

/* Sub-table columns*/
[SUB_TABLE_COLUMNS]

/* Sub-table content*/
[SUB_TABLE_ROW_VALUE]

Final Answer:
```

图 5.6　答案生成器提示模板

5.4　实验结果分析

5.4.1　实验设置

1. 数据集

本文使用两个广泛使用的公共数据集：**WikiTQ**[22]和 **TabFact**[24]进行基准测试。WikiTQ 是一个源自维基百科网页的公共表格问答数据集，没有预定义模板，需要对表格数据进行各种操作和推理来回答问题。TabFact 是一个基于表格的二元事实验证基准。其任务是根据表格确定给定陈述的真实性。在表格分解性能分析和消融实验中，本文还使用了 **HybridQA**[116]，这是一个集成了维基百科自由格式表格和文本的公共数据集，其复杂度相较 WikiTQ 更高，包含更多的大型表格。对于 WikiTQ 和 HybridQA，按照原文阐述基于完全匹配准确率报告各项工作性能；对于 TabFact 采用二元分类准确率报告各项工作性能。

2. 输入准备

本文遵循先前工作中广泛使用的方法[28,90]，将表格结构扁平化并转换为序列，纳入大语言模型的提示中。表格单元格用符号‘|’分隔，行用换行符分隔。问题直接包含在提示中，而答案仅用于模型评估。SQL 生成器中的大语言模型类似于 Text-to-SQL 的方法，结合问题、表格的表头信息以及上下文示例来构建提示，进而生成 SQL 查询。表格分解器利用生成的 SQL 查询和原始表格来提取子表格。而答案生成器中的大语言模型则将问题和子表格作为输入，构建一个提示，这样既能减少大语言模型的标记消耗，又能缩小其搜索空间。

3. 对比基线

用于比较的基线方法如下，主要分为基于文本推理和程序辅助推理两类：

- End-to-End QA：基础的端到端表格问答，将表格全部内容转为文本序列输入大语言模型；
- Few-Shot QA：将相关问题作为少样本提示上下文；
- Text-to-SQL：以生成 SQL 语句的执行结果作为表格问答结果；
- Chain-of-Thought[38]：运用思维链策略引导大语言模型逐步推理；
- TableCoT[117]：针对表格问答场景进行了思维链优化；
- Chain-of-Table[90]：定义表格操作引导表格分解过程；
- Tree-of-Table[109]：构建表格问答专用的推理结构树逐步求解；
- Dater[87]：分解子问题，并通过 SQL 得到各问题证据；
- Binder[64]：引导大语言模型生成 Python 可执行代码；
- TabSQLify[110]：引导大语言模型生成 SQL 拆分子表格

4. 实施细节

对于所有基于大语言模型的方法，包括 DesTab，使用 GPT-4o mini 和 GPT-3.5-turbo 作为基座模型。所有实验在配备 NVIDIA A100 80GB GPU 的服务器上运行。

5.4.2　主要结果

表 5.2　不同方法生成答案匹配准确率比较

方法	WikiTQ		TabFact	
	GPT35	GPT4m	GPT35	GPT4m
文本推理				
End-to-End QA	51.84	54.60	70.45	73.50
Few-Shot QA	52.56	57.60	71.54	75.10
Chain-of-Thought	53.48	58.82	65.37	77.24
TableCoT	52.40	58.94	73.14	80.26
Chain-of-Table	59.94	55.60	80.20	84.24
Tree-of-Table	61.11	58.20	81.92	85.28
程序辅助推理				
Text-to-SQL	52.90	53.40	64.71	66.34
Binder	56.74	58.86	79.17	84.63
Dater	52.81	58.33	78.01	80.98
TabSQLify	64.70	57.02	79.50	78.75
DesTab	**67.68(+2.98)**	**68.55(+9.69)**	**82.47(+0.55)**	**85.12(+0.49)**

表 5.2报告了整体性能结果。DesTab 在所有四个基准测试中始终优于所有竞争方案，准确率分别提高了 2.98%、9.69%、0.55% 和 0.49%。这证实了 DesTab 在处理复杂大型自由格式表格方面的有效性。在所有数据集中，WikiTQ 由于其表格内容的复杂性带来了最大的挑战。其中每个单元格都包含丰富的信息，常常整合了多个实体和冗长的描述性文本，这使得以往的基于程序辅助推理的方法无法适用。而 DesTab 仍能通过 SQL 为大型表格提取关键子表格，取得相较于所有程序辅助推理方法更优的结果。同时，相比于其他文本推理方法，DesTab 的开销更低，依靠 GPT-4o mini 强大的文本理解和推理能力展现出稳定的性能表现。

5.4.3　大型表格场景性能分析

表 5.3　不同表格规模下基线方法的性能比较

方法	表格规模 (Token)			平均值
	小型 (<2k)	中型 (2k ~ 4k)	大型 (>4k)	
Binder	56.54	26.13	6.41	29.69
Dater	62.50	42.34	34.62	46.49
Chain-of-Table	68.13	52.25	44.87	55.08
DesTab	**70.21(+2.08)**	**60.82(+8.57)**	**58.24(+13.37)**	**62.94(+7.86)**

为评估 DesTab 在不同表格规模下的可扩展性，本文根据标记（Token）数将

WikiTQ 数据集划分为三组：小型（<2k）、中型（2k-4k）和大型（>4k），并使用 GPT3.5-turbo 进行性能基准测试，如表 5.3 所示。DesTab 在所有表格规模下均持续优于现有方法。值得注意的是，当表格规模增大时，基线方法的性能显著下降，而 DesTab 表现出更强的稳定性——从中型到大型表格，其准确率仅略有下降。

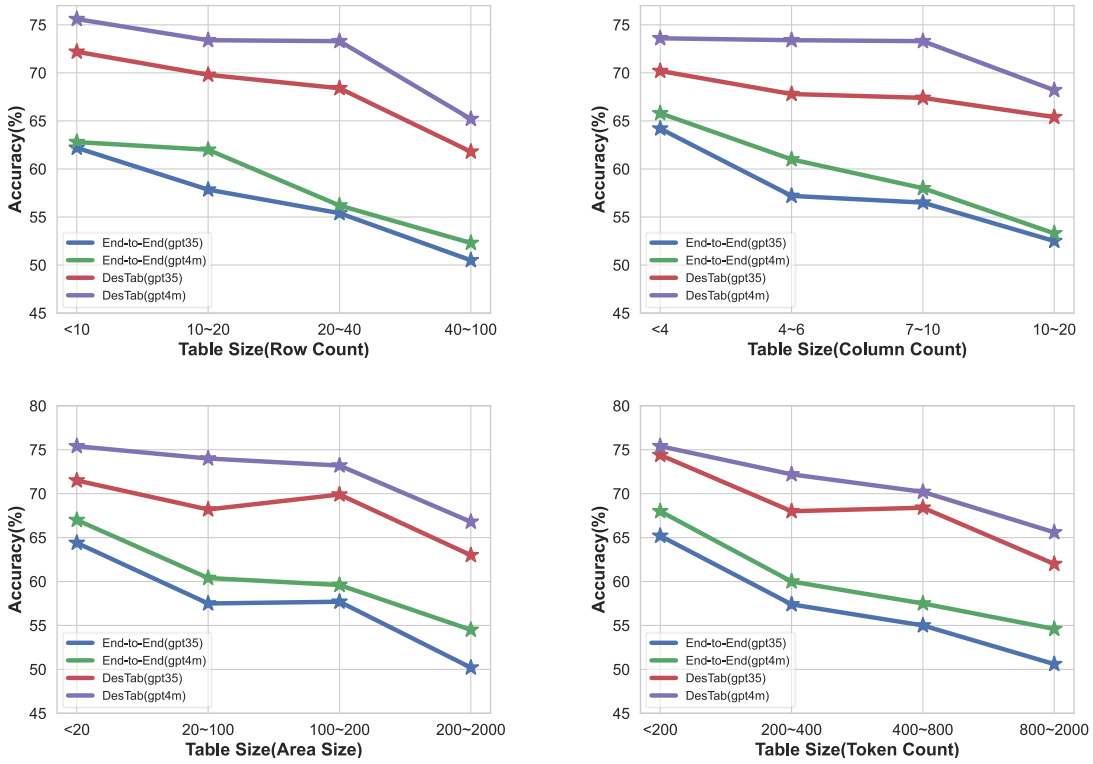

图 5.7　不同表格规模度量尺度下基线方法的性能比较

同时，本文通过设定行数、列数、区域大小和标记数量作为表格规模的度量，对各类方法性能变化的趋势进行了更细致的分析，如图 5.7所示。行数表示数据条目的数量，而列数反映每个条目的维度或属性的数量。区域大小是行计数和列计数的乘积，标记计数是从大语言模型的角度评估的表格大小。在每个度量标准中本文对比四种配置：基于 GPT-3.5-turbo 和 GPT-4o-mini 的端到端问答模型，以及与这两种模型结合的 DesTab 方法。实验结果表明，所有方案的准确率都随表格规模扩大而下降，同时不同方法对表格规模增长的适应能力有所差异：DesTab 与 GPT-4o-mini 结合后，在所有维度上均实现了最高准确率，并保持了相对平稳的性能曲线。这证明了 DesTab 在不同表格规模下的鲁棒性。

5.4.4　表格分解性能分析

接下来，本文评估了 DesTab 分解表格对输入开销产生的影响。鉴于 TabFact 中的问题较为简单，回答的初始准确率较高且几乎不含大型表格，本文选取 WikiTQ 和 HybirdQA 数据集中标记数目大于 4000 的表格记作 **WikiTQ-Large** 和 **HybridQA-Large**，如表 5.4所示。

表 5.4　WikiTQ 和 HybirdQA 筛选后数据集信息

数据集	训练集	验证集	测试集	平均标记数/表	平均标记数/答案
WikiTQ-Large	191	33	81	6860	2.9
HybridQA-Large	48541	2694	2630	9040	4.4

分别统计基于端到端表格问答方法和 DesTab 中的输入开销和最终答案准确率，结果如图 5.8所示，表格分解有效地将输入表格长度在两个数据集上分别减少了 69.8%、48.5%。这种大幅减少最小化了答案生成器的提示长度，使模型能够更有效地专注于重要数据，同时提高处理效率。同时两个数据集的准确率分别提高了 22.9%、1.2%，进一步表明大型表格问答的关键在于相关子表格的提取。

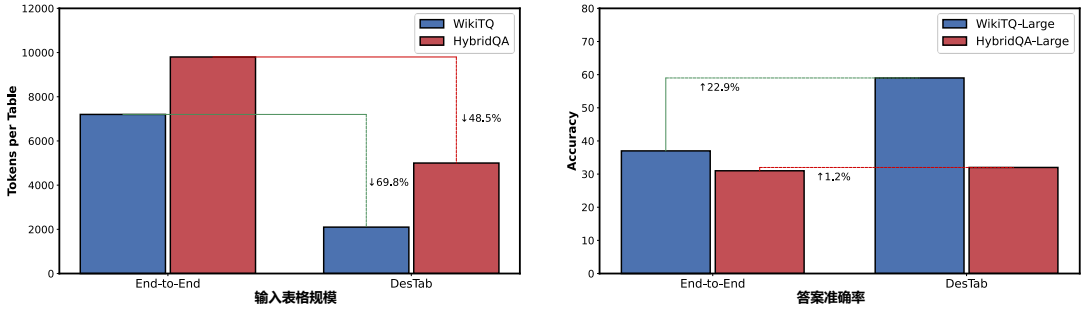

图 5.8　大型表格数据集上的输入开销和准确率比较

5.4.5　消融实验

本节评估 DesTab 框架中各个组件对模型整体性能的影响，在两个大型表格数据集上进行消融实验，使用四组方法变体：

- **w/o-表格清洗**：不进行表格信息的预处理清洗工作；
- **w/o-答案生成**：不采用 SQL 分解得到的子表格，而是直接执行 SQL 生成器生成的 SQL 查询作为结果；
- **w/o-SQL 生成和表格分解**：仅使用表格清洗和答案生成，退化至端到端表格问答方法；
- **w/o-SQL 验证**：不进行 SQL 验证，采用初次生成的 SQL 作为全局指导

表 5.5中结果表明，表格内容预处理可以有效减少噪声，提高生成答案准确率。直接执行生成的 SQL 查询会导致准确率大幅下降，因为 SQL 查询难以处理自由格式表格。然而当去除 SQL 生成任务或者 SQL 验证时，最终准确率出现明显下降，这表明 SQL 查询捕获了对表格分解有价值的结构信息，可以通过更准确的表格信息和执行验证提高生成的 SQL 查询质量，从而提高整体方法准确率。

表 5.5　DesTab 各组件消融实验

方法	准确率 (%)	
	WikiTQ-Large	HybridQA-Large
DesTab	59.26	31.83
w/o-表格清洗	55.02	30.40
w/o-答案生成	23.46	0.42
w/o-SQL 生成和表格分解	48.15	30.95
w/o-SQL 验证	57.40	28.24

5.5　本章小结

本章提出了 DesTab，一种用于大型自由格式表格的表格问答框架。DesTab 利用大语言模型（SQL 生成器）为每个输入问题生成 SQL 查询，利用 SQL 捕捉表格结构信息，对复杂大型表格的可靠推理提供了解决方案，同时避免了直接执行 SQL 查询获取答案的局限性。同时，为了提高生成 SQL 质量，本文引入了 SQL 验证器，可以根据分解过程动态调整指导全局分解的 SQL 模板。本章在两个公共数据集 WikiTQ 和 TabFact，以及两个衍生的大型表格数据集上评估了 DesTab 的性能。结果表明 DesTab 在答案生成的准确率方面均达到优秀水平。提升 Text-to-SQL 模型的性能表现不仅可以应用于数据库中的结构化表格，也可以在自由格式表格问答任务中发挥自己的作用。这意味着 Text-to-SQL 和表格问答拥有着共同的解决方案，二者都是智能化数据库系统中的关键要素，可以在今后的研究中互为参照。

第 6 章　总结与展望

6.1　本文工作

本文旨在提升基于大语言模型的 Text-to-SQL 性能，让缺乏数据库专业知识的普通用户可以使用自然语言与数据库进行交互，为智能数据库系统的自然语言接口设计安全可靠的代理。在自然语言处理与数据库系统的交叉研究领域，开发能够自动将自然语言问题转换为标准 SQL 的智能系统，是一个持续受到关注且极具挑战性的前沿方向。大语言模型有着丰富的领域知识和适用于不同任务的灵活性，成为实现这一目标的最佳选择。首先，本文需要提升 Text-to-SQL 算法面对复杂问题的推理能力，保证任务的较高准确率，这对提示工程中多智能体协作的框架和复杂问题的拆解策略提出了要求；然后，本文希望能将这种能力泛化到参数量级相对较小的模型，使其可以在各种算力受限的应用场景中部署，这需要重新审视微调策略与数据构成；最终，本文还致力于打通 Text-to-SQL 和表格问答任务的界限，将系统的应用范围扩展到人们的日常生活情景，这需要针对结构化表格和自由格式表格的差异对任务做出相应调整。

为解决上述挑战，本文首先以数据库中的结构化表格作为研究对象，深入探讨了 Text-to-SQL 的两种技术路线：基于提示工程和基于微调。

第一，从提示工程的角度出发，本文开展了关于智能体和推理策略设计的研究，提出了一种多智能体协作与模式对齐的 Text-to-SQL 算法 SA-SQL。本文首先制订了一套统一的，分预处理、推理和后处理三个模块处理的 Text-to-SQL 流程，以及模块之间的模式对齐策略。它通过对齐智能体的输入和输出，减少了指令跟随中的幻觉现象。此外，针对大语言模型在推理阶段面对复杂问题遇到的困难，本文将关系代数作为思维链（CoT）的中间语言，提高了思维链推理与 SQL 语法的契合度。SA-SQL 是一整套完善的框架体系，为将来进一步的多智能体协作研究奠定了良好基础。

第二，从微调的角度出发，本文开展了将大型大语言模型推理能力泛化到轻量化模型的研究，提出了一种数据增强和多任务协作的 Text-to-SQL 算法 EnhancedSQL。以往工作大多沿用公开数据集中的训练集在单一的 Text-to-SQL 任务上微调，这使得训练的轻量化模型严重缺乏鲁棒性和竞争优势。本文首先专注于数据增强，设计了在新域和大类错误进行数据生成的方法。接下来本文引入了模式链接和错误纠正任务，它们作为 SQL 相关任务可以增强大语言模型的 SQL 理解能力。最终，本文在轻量化模型上通过多任务协作提示实现了可与大型模型相媲美的推理性能，为将来 Text-to-SQL 在现实生活场景的部署落地展示了可行性。在这两种技术路径中，本文采用了多项研究策略，包括上下文学习、有监督

微调、少样本示例选择等，结合算法框架设计，在公开数据集上均取得了先进的性能，在多种场景中普遍优于现有方法，且具备更强的鲁棒性。

第三，为研究 Text-to-SQL 任务的适用范围和重要意义，本文将研究领域从结构化表格扩展到自由格式表格，针对目前大型自由格式表格输入开销大、推理困难的现状，设计了一个应用 Text-to-SQL 来进行大型自由格式表格问答任务的算法框架 DesTab。结果表明，即使 SQL 无法直接适用于自由格式表格，将其作为辅助分解中间件的算法性能也可以优于目前其他全表输入的算法，且实现了开销的显著降低。

这些研究成果展示了 Text-to-SQL 任务在表格与数据库中的重要地位，为未来的实践和应用提供了新的思路和方法。通过这些多角度的尝试，本文期望能够为智能化数据库系统带来更高效、更智能的解决方案，进一步推动该领域的科技进步。

6.2　展望

尽管本文对基于大语言模型的 Text-to-SQL 方法做了较为全面的分析，但纵观任务现状与发展趋势，仍有一些改进之处：

1. **新领域和开放域中的检索增强生成**：在第三章中，算法 SA-SQL 仅通过相似度计算选择了若干近似的 SQL 生成样例与数据集给出的提示作为上下文，没有引入额外的知识获取渠道，高度依赖大语言模型的先验知识。而现实生活的专业领域（如医学、金融等）包含大量专业术语或评价指标，Text-to-SQL 算法往往要和检索增强生成方法共同使用来获取专业知识，才能有效建立实体与数据库模式、问题需求与聚合计算的映射关系。
2. **更加高效的多任务协作模式**：在第四章中，算法 EnhancedSQL 需要使用多任务协作提示的方法，带来了额外的推理成本。同时，其性能距理论的上界值仍有一定差距。未来仍需对 SQL 相关综合数据和有监督微调、强化学习策略进一步研究，以减轻多任务协作可能带来的偏差与风险。
3. **文本推理和程序辅助推理的动态选择**：在第五章中，算法 DesTab 使用生成的 SQL 作为程序辅助完成表格分解子任务。在表格问答常见的两种推理模式中，文本推理使用较为自由但繁琐，适合逻辑推理；而程序辅助推理（如 Python、SQL）格式精炼但不灵活，适合检索与计算。针对自由格式表格问答，设计一种可以根据输入内容动态选择操作工具的算法仍是未来的一项挑战。它需要拥有更广泛的工具调用方式来处理表格问答中遇到的各类问题，如此方能将表格问答与 SQL 生成视为同一任务，打通结构化表格与生活中自由格式表格的界限。

参考文献

[1] 乔少杰, 李洲, 韩楠, 等. 人工智能赋能关系型数据库优化技术：现状与展望[J]. 计算机学报, 2025: 1-33.

[2] 刘雨蒙, 赵怡婧, 王碧聪, 等. 结构化数据库查询语言智能合成技术研究进展[J]. 计算机科学, 2024, 51(07): 40-48.

[3] 硕天鸾, 董一民. 人工智能时代数据库技术创新态势研究[J]. 信息通信技术与政策, 2024, 50(06): 17-22.

[4] DEVLIN J, CHANG M W, LEE K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[C]//Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers). 2019: 4171-4186.

[5] YU T, ZHANG R, ER H, et al. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases[C]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019: 1962-1979.

[6] LIU A, HU X, WEN L, et al. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability[A]. 2023.

[7] GAN Y, CHEN X, PURVER M. Exploring underexplored limitations of cross-domain text-to-sql generalization[C]//Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. 2021: 8926-8931.

[8] YU T, ZHANG R, YANG K, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task[C]//Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018: 3911-3921.

[9] LIN X V, SOCHER R, XIONG C. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing[C]//Findings of the Association for Computational Linguistics: EMNLP 2020. 2020: 4870-4888.

[10] DONG X, ZHANG C, GE Y, et al. C3: Zero-shot text-to-sql with chatgpt[A]. 2023.

[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, 2024, 36.

[12] GAO D, WANG H, LI Y, 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 B, REN C, YANG J, et al. Mac-sql: A multi-agent collaborative framework for text-to-sql[C]//Proceedings of the 31st International Conference on Computational Linguistics.

2025: 540-557.

[14] LEE D, PARK C, KIM J, et al. Mcs-sql: Leveraging multiple prompts and multiple-choice selection for text-to-sql generation[C]//Proceedings of the 31st International Conference on Computational Linguistics. 2025: 337-353.

[15] TAI C Y, CHEN Z, ZHANG T, et al. Exploring chain of thought style prompting for text-to-sql[C]//Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. 2023: 5376-5393.

[16] LI H, ZHANG J, LI C, et al. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql[C]//Proceedings of the AAAI Conference on Artificial Intelligence: Vol. 37. 2023: 13067-13075.

[17] LI J, HUI B, CHENG R, et al. Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing[C]//Proceedings of the AAAI Conference on Artificial Intelligence: Vol. 37. 2023: 13076-13084.

[18] LI H, ZHANG J, LIU H, 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.

[19] YANG J, HUI B, YANG M, et al. Synthesizing text-to-sql data from weak and strong llms[C]//Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2024: 7864-7875.

[20] GU Z, FAN J, TANG N, et al. Interleaving pre-trained language models and large language models for zero-shot nl2sql generation[A]. 2023.

[21] ROZIERE B, GEHRING J, GLOECKLE F, et al. Code llama: Open foundation models for code[A]. 2023.

[22] PASUPAT P, LIANG P. Compositional semantic parsing on semi-structured tables[C]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2015: 1470-1480.

[23] YIN P, NEUBIG G, YIH W T, et al. Tabert: Pretraining for joint understanding of textual and tabular data[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020: 8413-8426.

[24] CHEN W, WANG H, CHEN J, et al. Tabfact: A large-scale dataset for table-based fact verification[C]//International Conference on Learning Representations. 2020.

[25] HERZIG J, NOWAK P K, MUELLER T, et al. Tapas: Weakly supervised table parsing via pre-training[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020: 4320-4333.

[26] BAO J, TANG D, DUAN N, et al. Table-to-text: Describing table region with natural lan-

guage[C]//Proceedings of the AAAI conference on artificial intelligence: Vol. 32. 2018.

[27] ZHANG H, SI S, ZHAO Y, et al. Opent2t: An open-source toolkit for table-to-text generation[C]//Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. 2024: 259-269.

[28] LIU T, WANG F, CHEN M. Rethinking tabular data understanding with large language models[C]//Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). 2024: 450-482.

[29] ZHONG V, XIONG C, SOCHER R. Seq2sql: Generating structured queries from natural language using reinforcement learning[A]. 2017.

[30] SHI L, TANG Z, ZHANG N, et al. A survey on employing large language models for text-to-sql tasks[A]. 2024.

[31] SUTSKEVER I, VINYALS O, LE Q V. Sequence to sequence learning with neural networks [J]. Advances in neural information processing systems, 2014, 27.

[32] JHA D, WARD L, YANG Z, et al. Irnet: A general purpose deep residual regression framework for materials discovery[C]//Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019: 2385-2393.

[33] LI J, HUI B, QU G, et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls[J]. Advances in Neural Information Processing Systems, 2024, 36.

[34] ZHAO W X, ZHOU K, LI J, et al. A survey of large language models[A]. 2023.

[35] WEI J, TAY Y, BOMMASANI R, et al. Emergent abilities of large language models[A]. 2022.

[36] BROWN T, MANN B, RYDER N, et al. Language models are few-shot learners[J]. Advances in neural information processing systems, 2020, 33: 1877-1901.

[37] GAO Y, XIONG Y, GAO X, et al. Retrieval-augmented generation for large language models: A survey[A]. 2023.

[38] WEI J, WANG X, SCHUURMANS D, et al. Chain-of-thought prompting elicits reasoning in large language models[J]. Advances in neural information processing systems, 2022, 35: 24824-24837.

[39] ZHOU D, SCHÄRLI N, HOU L, et al. Least-to-most prompting enables complex reasoning in large language models[A]. 2022.

[40] YAO S, YU D, ZHAO J, et al. Tree of thoughts: Deliberate problem solving with large language models[J]. Advances in Neural Information Processing Systems, 2024, 36.

[41] HU E J, WALLIS P, ALLEN-ZHU Z, et al. Lora: Low-rank adaptation of large language

models[C]//International Conference on Learning Representations. 2022.

[42] WEI J, BOSMA M, ZHAO V, et al. Finetuned language models are zero-shot learners[C]// International Conference on Learning Representations. 2022.

[43] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[44] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[45] BOGIN B, BERANT J, GARDNER M. Representing schema structure with graph neural networks for text-to-sql parsing[C]//Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019: 4560-4565.

[46] FU Y, BAILIS P, STOICA I, et al. Break the sequential dependency of llm inference using lookahead decoding[C]//Proceedings of the 41st International Conference on Machine Learning. 2024: 14060-14079.

[47] PAN L, SAXON M, XU W, et al. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies[J]. Transactions of the Association for Computational Linguistics, 2024, 11: 484-506.

[48] WANG X, WEI J, SCHUURMANS D, et al. Self-consistency improves chain of thought reasoning in language models[C]//The Eleventh International Conference on Learning Representations. 2023.

[49] XIA H, JIANG F, DENG N, et al. *r*^3:" this is my sql, are you with me?" a consensus-based multi-agent system for text-to-sql tasks[A]. 2024.

[50] LI Z, WANG X, ZHAO J, et al. Pet-sql: A prompt-enhanced two-stage text-to-sql framework with cross-consistency[A]. 2024.

[51] RAJKUMAR N, LI R, BAHDANAU D. Evaluating the text-to-sql capabilities of large language models[A]. 2022.

[52] CHEN X, LIN M, SCHÄRLI N, et al. Teaching large language models to self-debug[C]//The Twelfth International Conference on Learning Representations. 2024.

[53] ZHANG H, CAO R, CHEN L, 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. 2023: 3501-3532.

[54] SUI G, LI Z, LI Z, et al. Reboost large language model-based text-to-sql, text-to-python, and text-to-function–with real applications in traffic domain[A]. 2023.

[55] ARORA A, BHAISAHEB S, PATWARDHAN M, et al. A generic prompt for an llm that enables nl-to-sql across domains and compositions[C]//Proceedings of the Eleventh International Conference on Learning Representations, Kigali, Rwanda. 2023: 1-5.

[56] XIE Y, JIN X, XIE T, et al. Decomposition for enhancing attention: Improving llm-based text-to-sql through workflow paradigm[C]//Findings of the Association for Computational Linguistics ACL 2024. 2024: 10796-10816.

[57] CHEN X, WANG T, QIU T, et al. Open-sql framework: Enhancing text-to-sql on open-source large language models[A]. 2024.

[58] CHANG S, FOSLER-LUSSIER E. How to prompt llms for text-to-sql: A study in zero-shot, single-domain, and cross-domain settings[C]//NeurIPS 2023 Second Table Representation Learning Workshop. 2023.

[59] FAN Y, HE Z, REN T, et al. Metasql: A generate-then-rank framework for natural language to sql translation[C]//2024 IEEE 40th International Conference on Data Engineering (ICDE). 2024: 1765-1778.

[60] ZHANG T, CHEN C, LIAO C, et al. Sqlfuse: Enhancing text-to-sql performance through comprehensive llm synergy[A]. 2024.

[61] TALAEI S, POURREZA M, CHANG Y C, et al. Chess: Contextual harnessing for efficient sql synthesis[A]. 2024.

[62] ARORA A, BHAISAHEB S, NIGAM H, et al. Adapt and decompose: Efficient generalization of text-to-sql via domain adapted least-to-most prompting[C]//Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP. 2023: 25-47.

[63] GAN Y, CHEN X, XIE J, et al. Natural sql: Making sql easier to infer from natural language specifications[C]//Findings of the Association for Computational Linguistics: EMNLP 2021. 2021: 2030-2042.

[64] CHENG Z, XIE T, SHI P, et al. Binding language models in symbolic languages[C]//The Eleventh International Conference on Learning Representations. 2023.

[65] ZHANG Q, DONG J, CHEN H, et al. Structure guided large language model for sql generation[A]. 2024.

[66] LIU X, TAN Z. Divide and prompt: Chain of thought prompting for text-to-sql[A]. 2023.

[67] ZHANG H, CAO R, XU H, et al. Coe-sql: In-context learning for multi-turn text-to-sql with chain-of-editions[C]//Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). 2024: 6487-6508.

[68] GUO C, TIAN Z, TANG J, et al. Prompting gpt-3.5 for text-to-sql with de-semanticization and skeleton retrieval[C]//Pacific Rim International Conference on Artificial Intelligence. 2023: 262-274.

[69] GUO C, TIAN Z, TANG J, et al. Retrieval-augmented gpt-3.5-based text-to-sql framework with sample-aware prompting and dynamic revision chain[C]//International Conference on

Neural Information Processing. 2023: 341-356.

[70] REN T, FAN Y, HE Z, et al. Purple: Making a large language model a better sql writer[C]// 2024 IEEE 40th International Conference on Data Engineering (ICDE). 2024: 15-28.

[71] GUO C, TIAN Z, TANG J, et al. A case-based reasoning framework for adaptive prompting in cross-domain text-to-sql[A]. 2023.

[72] SUN R, ARIK S O, NAKHOST H, et al. Sql-palm: Improved large language modeladaptation for text-to-sql[A]. 2023.

[73] NI A, IYER S, RADEV D, et al. Lever: Learning to verify language-to-code generation with execution[C]//International Conference on Machine Learning. 2023: 26106-26128.

[74] ACHIAM J, ADLER S, AGARWAL S, et al. Gpt-4 technical report[A]. 2023.

[75] XUE S, JIANG C, SHI W, et al. Db-gpt: Empowering database interactions with private large language models[A]. 2023.

[76] ROZIERE B, GEHRING J, GLOECKLE F, et al. Code llama: Open foundation models for code[A]. 2023.

[77] THORPE D G, DUBERSTEIN A J, KINSEY I A. Dubo-sql: Diverse retrieval-augmented generation and fine tuning for text-to-sql[A]. 2024.

[78] ZHANG C, MAO Y, FAN Y, et al. Finsql: model-agnostic llms-based text-to-sql framework for financial analysis[C]//Companion of the 2024 International Conference on Management of Data. 2024: 93-105.

[79] 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. 2024: 8212-8220.

[80] HONG Z, YUAN Z, CHEN H, et al. Knowledge-to-sql: Enhancing sql generation with data expert llm[C]//Findings of the Association for Computational Linguistics ACL 2024. 2024: 10997-11008.

[81] LIU X, JI K, FU Y, et al. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks[C]//Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2022: 61-68.

[82] DETTMERS T, PAGNONI A, HOLTZMAN A, et al. Qlora: Efficient finetuning of quantized llms[J]. Advances in Neural Information Processing Systems, 2024, 36.

[83] LIU H, TAM D, MUQEETH M, et al. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning[J]. Advances in Neural Information Processing Systems, 2022, 35: 1950-1965.

[84] LESTER B, AL-RFOU R, CONSTANT N. The power of scale for parameter-efficient prompt tuning[C]//Proceedings of the 2021 Conference on Empirical Methods in Natural Language

Processing. 2021: 3045-3059.

[85] JANG J, KIM S, YE S, et al. Exploring the benefits of training expert language models over instruction tuning[C]//Proceedings of the 40th International Conference on Machine Learning. 2023: 14702-14729.

[86] WANG Y, QI J, GAN J. Accurate and regret-aware numerical problem solver for tabular question answering[C]//Proceedings of the AAAI Conference on Artificial Intelligence: Vol. 39. 2025: 12775-12783.

[87] YE Y, HUI B, YANG M, et al. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning[C]//Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval. 2023: 174-184.

[88] LEE Y, KIM S, ROSSI R A, et al. Learning to reduce: Towards improving performance of large language models on structured data[J]. ICML Workshop, 2024.

[89] PATNAIK S, CHANGWAL H, AGGARWAL M, et al. CABINET: Content relevance-based noise reduction for table question answering[C]//The Twelfth International Conference on Learning Representations. 2024.

[90] WANG Z, ZHANG H, LI C L, et al. Chain-of-table: Evolving tables in the reasoning chain for table understanding[C]//International Conference on Learning Representations. 2024.

[91] WANG Z, DONG H, JIA R, et al. Tuta: Tree-based transformers for generally structured table pre-training[C]//Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021: 1780-1790.

[92] LIU Q, CHEN B, GUO J, et al. TAPEX: Table pre-training via learning a neural SQL executor [C]//International Conference on Learning Representations. 2022.

[93] LEWIS M, LIU Y, GOYAL N, et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020: 7871-7880.

[94] JIANG Z, MAO Y, HE P, et al. Omnitab: Pretraining with natural and synthetic data for few-shot table-based question answering[C]//Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2022: 932-942.

[95] YANG H, YUE S, HE Y. Auto-gpt for online decision making: Benchmarks and additional opinions[A]. 2023.

[96] WU Q, BANSAL G, ZHANG J, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation[C]//ICLR 2024 Workshop on Large Language Model (LLM) Agents. 2024.

[97] MA P, ZHUANG X, XU C, et al. Sql-r1: Training natural language to sql reasoning model by reinforcement learning[A]. 2025.

[98] KATSOGIANNIS-MEIMARAKIS G, KOUTRIKA G. A deep dive into deep learning approaches for text-to-sql systems[C]//Proceedings of the 2021 International Conference on Management of Data. 2021: 2846-2851.

[99] XIAO C, DYMETMAN M, GARDENT C. Sequence-based structured prediction for semantic parsing[C]//Annual meeting of the Association for Computational Linguistics (ACL). 2016: 1341-1350.

[100] QIN Y, CHEN C, FU Z, et al. Route: Robust multitask tuning and collaboration for text-to-sql [A]. 2024.

[101] GAN Y, CHEN X, HUANG Q, et al. Towards robustness of text-to-sql models against synonym substitution[C]//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). 2021: 2505-2515.

[102] DENG X, HASSAN A, MEEK C, et al. Structure-grounded pretraining for text-to-sql[C]// Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2021: 1337-1350.

[103] TOUVRON H, LAVRIL T, IZACARD G, et al. Llama: Open and efficient foundation language models[A]. 2023.

[104] ZHENG Y, ZHANG R, ZHANG J, et al. Llamafactory: Unified efficient fine-tuning of 100+ language models[C]//Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations). 2024: 400-410.

[105] JIANG A Q, SABLAYROLLES A, MENSCH A, et al. Mistral 7b[A]. 2023.

[106] CAFARELLA M J, HALEVY A, WANG D Z, et al. WebTables: Exploring the power of tables on the web[J]. Proceedings of the VLDB Endowment, 2008, 1(1): 538-549.

[107] JIN N, SIEBERT J, LI D, et al. A survey on table question answering: recent advances[C]// China Conference on Knowledge Graph and Semantic Computing. 2022: 174-186.

[108] LIU N F, LIN K, HEWITT J, et al. Lost in the middle: How language models use long contexts [J]. Transactions of the Association for Computational Linguistics, 2024, 12: 157-173.

[109] JI D, ZHU L, GAO S, et al. Tree-of-table: Unleashing the power of llms for enhanced large-scale table understanding[A]. 2024.

[110] NAHID M, RAFIEI D. Tabsqlify: Enhancing reasoning capabilities of llms through table decomposition[C]//Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). 2024: 5725-5737.

[111] ZHANG S, TUAN L A, ZHAO C. Syntqa: Synergistic table-based question answering via mixture of text-to-sql and e2e tqa[C]//Findings of the Association for Computational Linguistics: EMNLP 2024. 2024: 2352-2364.

[112] LU W, ZHANG J, FAN J, et al. Large language model for table processing: A survey[J]. Frontiers of Computer Science, 2025, 19(2): 192350.

[113] SUI Y, ZOU J, ZHOU M, et al. Tap4llm: Table provider on sampling, augmenting, and packing semi-structured data for large language model reasoning[C]//Findings of the Association for Computational Linguistics: EMNLP 2024. 2024: 10306-10323.

[114] POURREZA M, LI H, SUN R, et al. Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql[A]. 2024.

[115] DATAR M, IMMORLICA N, INDYK P, et al. Locality-sensitive hashing scheme based on p-stable distributions[C]//Proceedings of the twentieth annual symposium on Computational geometry. 2004: 253-262.

[116] CHEN W, ZHA H, CHEN Z, et al. Hybridqa: A dataset of multi-hop question answering over tabular and textual data[C]//Findings of the Association for Computational Linguistics: EMNLP 2020. 2020: 1026-1036.

[117] CHEN W. Large language models are few (1)-shot table reasoners[C]//Findings of the Association for Computational Linguistics: EACL 2023. 2023: 1120-1130.