arXiv:2602.12574v1 [cs.DB] 13 Feb 2026

结合推理路径优化的蒙特卡洛树搜索：面向对话式文本到 NoSQL 的小语言模型

Xubang Xiong[1], Raymond Chi-Wing Wong[1], Yuanfeng Song[1]香港科技大学，香港，中国

摘要

NoSQL 数据库因其灵活性和可扩展性，已广泛应用于大数据分析、地理空间应用和医疗保健服务。然而，查询 NoSQL 数据库需要专业的技术知识，为用户设置了较高的门槛。尽管近期研究已探索文本到 NoSQL 问题，但其主要聚焦于单轮交互，忽略了现实世界查询的对话性质。为弥合这一差距，我们提出了对话式文本到 NoSQL 任务：给定一个自然语言问题、一个 NoSQL 数据库和对话历史，生成 NoSQL 查询。为解决这一任务，我们提出 Stage-MCTS，这是一种通过将查询生成形式化为搜索问题，赋予小型语言模型（SLM）NoSQL 特定推理能力的框架。该框架采用由基于规则的奖励引导的蒙特卡洛树搜索（MCTS）来生成逐步推理数据，随后进行渐进式监督微调（SFT）和自训练策略。我们进一步构建了 CoNoSQL，这是一个跨领域数据集，包含超过 2,000 个对话和 150 个数据库，用于支持评估。实验表明，我们的方法优于先进水平的大型推理模型，使执行值匹配（EVM）准确率最高提升 7.93%。

1 引言

非关系型数据库（NoSQL）是一种先进的数据存储与管理系统，以非表格形式存储数据，旨在存储没有预定义结构的数据。由于它能够灵活地处理大规模数据，性能出色，因此在大数据分析 [21, 38], 地理空间应用 [2, 15] 和医疗保健服务 [51, 58, 67], 领域备受重视。

尽管 NoSQL 数据库有诸多优势，用户仍需学习专门知识才能编写查询语句并有效管理数据，这构成了很高的技术门槛。为应对这一挑战，研究者提出了使用自然语言接口查询 NoSQL 数据库的方法，使用户与 NoSQL 数据库之间的交互更加直观。现有研究 [36, 46] 探讨了文本到 NoSQL 问题。

给定自然语言问题和 NoSQL 数据库信息，系统需要预测 NoSQL 查询。然而，本研究假设，在单轮交互场景中，只能将一个自然语言问题映射为最终的 NoSQL 查询，以满足用户的需求。

在实际应用中，现实场景很少涉及孤立的查询。相反，大多数实际交互都由连续的多轮交流组成。为了实现有效的人机通信系统，必须考虑对话上下文，以细致入微的方式理解先前的交流，并迭代地完善目标。因此，单轮交互范式存在明显局限，限制了对 NoSQL 数据库自然语言

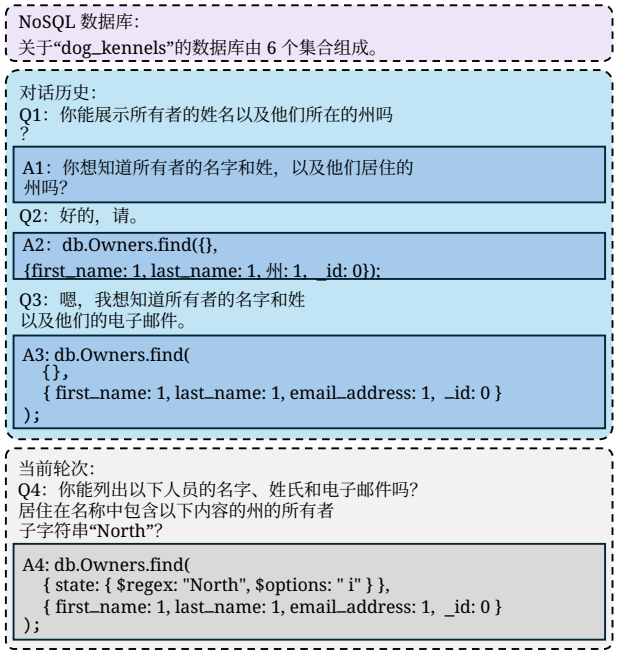


图 1：CoNoSQL 数据集中的一段对话。在此场景中，MongoDB 被用作 NoSQL 数据库的代表。Q_i 表示第 i 轮中的用户问题，A_i 表示系统的响应（即通常为 NoSQL 查询）。

接口固有高级能力的充分利用。受此启发，我们提出了**对话式文本到 NoSQL** 问题。给定一个自然语言问题、相应的 NoSQL 数据库信息以及对话历史，系统需要生成满足用户需求的 NoSQL 查询。

如图 1 所示，给定当前轮次的用户问题，系统会利用 NoSQL 数据库中关于“dog_kennels”的信息，以及所有与上下文相关的对话历史来生成 NoSQL 查询。这表明，本文研究如何增强带有自然语言接口的 NoSQL 数据库，使其能够结合数据库模式和当前对话上下文，生成针对用户所请求的特定信息量身定制的 NoSQL 查询。

不同于传统的 SQL 查询，NoSQL 查询依赖聚合流水线进行数据处理。此外，NoSQL 数据库支持动态模式，这意味着同一集合中的文档可以具有不同的结构。这些独特特征使对话式文本到 NoSQL 问题极具挑战性。为了解决这一问题，同时兼顾效率与效果，一种直接的方法是部署小型语言模型（SLMs），在思维链（CoT） [24, 64] 框架下生成 NoSQL 查询。思维链推理已被

证明能够在包括数学推理 [10, 73] 和科学问题求解 [55, 63]在内的多种任务中显著提升语言模型的性能。然而，附录表5中的初步实验表明，即使采用思维链推理，SLMs 的表现仍不如不使用思维链推理的方法。这一现象可归因于 SLMs 固有的 NoSQL 专属推理能力不足。

受这些观察结果的启发，我们提出了*Stage-MCTS*，这是一种旨在赋予小语言模型原生 NoSQL 专项推理能力的框架。具体而言，我们将对话式文本到 NoSQL 问题建模为搜索问题。小语言模型首先逐步生成可执行的 NoSQL 阶段，每个阶段都配有相应的自然语言注释，然后将这些预先生成的阶段组装起来，生成最终查询。为获取大量高质量训练数据，我们利用经蒙特卡洛树搜索（MCTS）增强的小语言模型，在基于规则的奖励模型的引导下生成多条推理路径。MCTS 将复杂的对话式文本到 NoSQL 任务分解为更易处理的单步生成子任务。此外，其固有的逐步生成机制天然有利于提取步骤级训练数据。随后，我们采用所提出的基于奖励的采样和路径优化策略对生成的路径进行筛选，以确保数据质量。构建训练数据集后，小语言模型会进行渐进式监督微调（SFT），即三阶段监督微调，旨在使小语言模型逐步从简单到复杂地学习知识。鉴于小语言模型在专门推理方面固有的局限性，我们采用多轮自训练流水线，迭代生成高质量数据，并以多样化的推理路径丰富训练集。经过多轮自训练后，更新后的小语言模型采用基于蒙特卡洛树搜索的测试时扩展方法来预测 NoSQL 查询，并由广受认可的自监督奖励模型 [27, 41]引导查询生成。

为验证所提出框架的有效性，我们构建了一个跨领域对话式文本到 NoSQL 数据集 CoNoSQL，用于专门的实验评估。该数据集包含 2,000 多段对话、9,000 多轮对话交互以及 150 多个跨领域数据库。CoNoSQL 涵盖广泛的真实世界应用场景，因此能够有效确保实验结果的泛化能力和可靠性。

本研究的主要贡献总结如下。

- 据我们所知，这是首项提出对话式文本到 NoSQL 任务，并将 NoSQL 查询生成建模为搜索问题的研究。
- 所提出的 Stage-MCTS 框架已被证明能够有效赋予小型语言模型现有车型所缺乏的、针对 NoSQL 定制的推理能力。
- 我们构建了一个名为 CoNoSQL 的新数据集，用于验证 Stage-MCTS 的有效性。该数据集结合了大语言模型与人工评估的协作，既提高了数据集整理的效率，又确保生成高质量、可靠的数据。
- 大量模拟验证了 Stage-MCTS 的可行性，对性能和消融研究的分析也为如何使用该框架提供了指导。值得注意的是，Stage-MCTS 在执行值匹配（EVM）准确率方面优于 DeepSeek-R1 等先进推理模型，以及 GPT-4o，最高提升达 7.93%。

本文其余部分安排如下：第2节介绍必要的预备知识。第3节详细阐述所提出框架的设计与分析，以及验证数据集的构建。第4节展示所提出框架的性能。最后，第5节回顾相关研究，第6节总结本文的主要发现。

2 预备知识

2.1 任务

对话式文本到 NoSQL 任务的形式化定义如下。在每个交互轮次 i，给定当前用户问题 q_i、NoSQL 数据库信息（例如模式）D 和对话历史 $H_i = \{(q_1, y_1)...(q_{i-1}, y_{i-1})\}$，车型生成 NoSQL 查询 y_i，以满足用户的需求。

2.2 NoSQL 查询语言

NoSQL 数据库与 SQL 数据库在查询范式上存在根本差异，这源于它们底层的数据模型和设计理念不同。

两种查询语言的核心差异在于其查询执行阶段。以 MongoDB 为例，它采用聚合管道，如图 2 所示。该流水线由多个阶段组成。每个阶段都会处理传入的文档，并将转换后的结果传递给下一阶段。在此示例中，$*group* 运算符作用于已由 $*match* 运算符筛选的数据，而非直接处理存储在 “orders” 集合中的原始数据。

```
db.orders.aggregate ([
    { $match: { orderDate: { $gte: ISO日期 ("2023年 -01 -01"),
$lt:  ISODate ( "2024 01 01 ) } } },
    { $group: { _id:          "$客户ID",          总销售额: { $求和:          "
$金额" } } },
    { $排序: { 总销售额: -1 } }
])
```

图 2：NoSQL 查询示例。

3 方法

3.1 概述

利用小语言模型（SLM）探索生成更高质量训练数据的方法。为增强 NoSQL 推理能力，我们提出了一种分阶段增强的思维链（CoT）数据增强方法，详见第 3.2 节。如图 3 所示，集成了蒙特卡洛树搜索（MCTS）的小语言模型在基于规则的奖励模型的指导下生成多条推理路径。随后，通过基于奖励的采样，从这些路径中选出前 k 条正确路径。如果在某个采样步骤中没有纠正出有效的推理路径，则使用真实值对错误路径进行修正。之后，这些已收集的推理路径会被纳入训练集，用于进行三阶段微调，详见第 3.3 节。经过微调的模型随后用于在下一轮迭代中生成更高质量的训练数据，详见第 3.4 节。最后，第 3.5 节介绍如何利用自我改进的小语言模型，通过基于 MCTS 的测试时扩展来预测 NoSQL 查询。为验证所

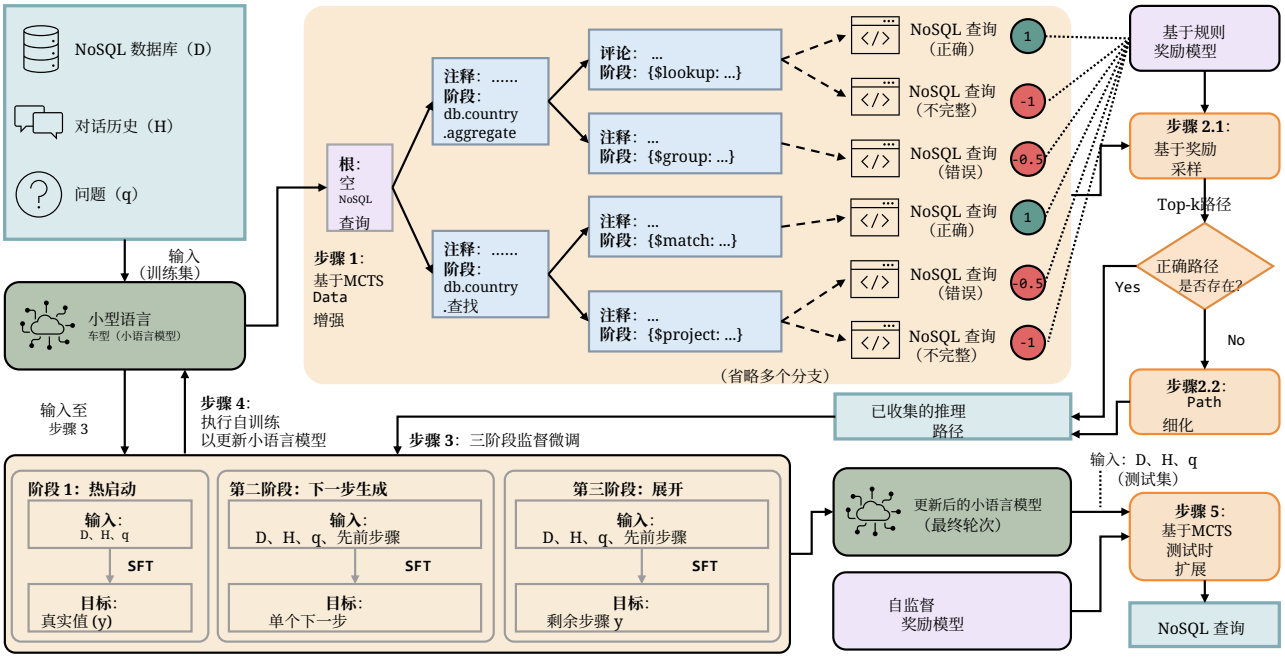


图 3：采用小语言模型（SLM）和基于规则的奖励模型生成多条推理路径。通过基于奖励的采样，从这些路径中选出 top-k 条正确路径。如果某次采样未能得到正确路径，则利用真实值对错误路径进行优化。随后，使用已收集的推理路径迭代训练 SLM。最后，基于蒙特卡洛树搜索（MCTS）进行测试时扩展，并使用自训练的 SLM 预测 NoSQL 查询。

提方法的有效性，我们构建了 CoNoSQL——一个跨领域对话式文本到 NoSQL 数据集，用于评估。

3.2 阶段增强的思维链生成

3.2.1 将文本到 *NoSQL* 构建为搜索问题。 对话式文本到 NoSQL 任务可定义为在庞大的潜在阶段空间中进行搜索。与 SQL 查询不同，NoSQL 查询（例如 MongoDB 查询）包含多个阶段，这些阶段嵌入在 aggregate、find、distinct 和 count 等方法中，详见第 2.2 节。

因此，给定一个 NoSQL 数据库 D、对话历史 H_i 和当前问题 q_i，搜索空间 S 由这些阶段的所有可能组合构成。该空间表示为树结构 $\Phi = (V, E)$，其中：

节点（V）。每个节点 $v \in V$ 表示查询构建过程中某个特定步骤对应的部分推理状态，如图 4 所示。具体而言，根节点 v_0 表示初始的空查询，并封装了输入信息，包括 NoSQL 数据库 D、对话历史 H_i 和当前问题 q_i。中间节点（例如 v_1）存储一条与生成阶段相关联的注释，该阶段有助于构建最终的 NoSQL 查询。叶节点 v_d（此处以 v_3 为例）表示一个已完整构建的 NoSQL 查询或单个生成阶段。

边（E）。每条边 $e \in E$ 表示查询构建过程中的一个操作，例如生成一个阶段或构建最终的 NoSQL 查询。这些操作模拟搜索树中各个中间查询状态之间的转换。

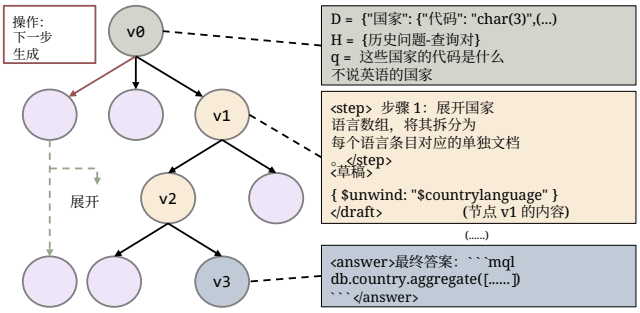


图 4：将文本到 NoSQL 表述为搜索问题

在搜索树中，任何以终端节点为终点的根到叶路径都会形成一条轨迹 $t = v_0 \oplus v_1 \oplus \cdots \oplus v_d$，其中 $\oplus$ 表示沿路径各条边所关联操作的串联。

3.2.2 阶段增强思维链。 以往方法主要生成自然语言思维链 [24, 64]。然而，在对话式文本到 NoSQL 任务中，推理步骤需要根据 NoSQL 特定推理的独特特点进行定制，从而帮助车型更好地掌握任务要求。为此，我们提出了阶段增强思维链。如图 4 所示，每个推理步骤都对应一个可执行的 NoSQL 阶段，并配有自然语言注释。作为最终查询不可或缺的组成部分，这种设计有效提升了可解释性。

3.2.3 用于数据增强的改进的蒙特卡洛树搜索。 本研究旨在训练一个具备推理能力的小语言模型，以完成对话式文本到NoSQL 任务。具体而言，小语言模型首先逐步生成可执行的NoSQL 阶段，并为每个阶段配上相应的自然语言注释。随后，它将这些预先生成的阶段组合起来，输出最终查询。为解决该方法需要大量包含详细推理步骤的训练数据这一限制，我们提出利用蒙特卡洛树搜索。首先，它将复杂的对话式文本到NoSQL 任务拆解为更易处理的单步生成子任务，从而有效降低整体问题的复杂度。此外，基于我们提出的框架（该框架将文本到 NoSQL 表述为搜索问题），蒙特卡洛树搜索所固有的逐步生成机制可自然地用于推导步骤级训练数据。这一双重优势不仅简化了核心生成任务，还提供了结构化、细粒度的数据，以提升车型训练效果和泛化能力。改进的蒙特卡洛树搜索算法详情如下。

节点选择。 采用众所周知的树的置信上界（UCT） [23]，从候选节点中选择最佳节点，即 UCT(v) 值最大的节点。此选择过程在数学上表示为：

$$\mathrm{UCT}(v) = Q(v) + c\sqrt{\frac{\ln N_{\text{parent}}(v)}{N(v)}}; \quad \text{where} \quad Q(v) = \frac{q(v)}{N(v)}. \tag{1}$$

在此过程中， $N(v)$ 表示对节点 v 的访问次数，而 N父节点（v）表示对 v 的父节点的访问次数。预测奖励 $q(v)$ 是奖励模型的输出，将在下一节中讨论，并通过反向传播进行更新。 c 是一个用于平衡利用与探索的常数。

阶段扩展。 给定所选节点 v_{select}，小语言模型生成 m 个子节点，其中 m 是一个正整数超参数。每个子节点包含一个阶段及其对应的推理，这些内容源自存储在 v_{select} 中的先前步骤。

基于阈值的推演。 完整的推演过程包括反复进行节点选择和阶段扩展，直到到达终止节点。然而，复杂的 NoSQL 查询可能涉及多个阶段，在训练数据采样期间生成这些阶段可能会耗费大量时间。为平衡蒙特卡洛树搜索的效率和效果，我们提出了一个基于阈值的推演步骤。具体而言，当所选节点的深度 v_{select} 达到预定义的正整数阈值 max_depth时，策略小语言模型会在单次响应中生成所有剩余阶段和最终的 NoSQL 查询。整个推演过程中，所有新扩展的子节点都会持续保留在树结构中。

反向传播。 在终止节点处，反向传播从评估执行值匹配（*EVM*）指标开始，该指标基于预测的 NoSQL 查询和真实值 NoSQL 查询计算。EVM 评估预测查询的执行结果中的值是否与真实查询中的值匹配，因为它消除了自定义字段名称的干扰，只关注唯一值。EVM 的详细定义见附录 C.2 节。考虑从根节点 v_0 到终止节点 v_d 的轨迹 $t = v_0 \oplus v_1 \oplus \cdots \oplus v_d$。对于该轨迹上的每个节点$v$，预测奖励 $q(v)$ 和访问计数 $N(v)$ 均按如下方式更新： $q(v) = q(v) + r$, $N(v) = N(v) + 1$。奖励 r定义为：若 EVM 为真，则为 1。0；若 EVM 为假，则为 -0.5；若预测的 NoSQL 查询无法执行，则为 -1.0。对于无法执行的预测 NoSQL 查询，奖励为该值。这种广泛采用的基于规则的奖励模型 [16, 57] 既准确又高效。

3.2.4 基于奖励的采样与路径优化。 经过 $N_{rollout}$次展开后，会收集到多条完整的推理轨迹，形成一组候选 NoSQL 查询推理轨迹 $\mathrm{T} = \{t_1, \ldots\ldots, t_n\}$。其中， $N_{rollout}$ 是用户定义的超参数。在这些候选轨迹中，我们筛选并保留奖励为 1.0 的前 k 条正确路径。如果某一生成轮次未能得出有效的推理路径，则利用小语言模型，根据真实查询对错误路径进行优化。具体而言，给定一个 NoSQL 数据库 D、对话历史 H_i、当前问题 q_i 和真实查询 y_i，小语言模型会在单次响应中生成所有推理路径。通过小语言模型进行路径优化的提示词见附录中的图8。

3.3 三阶段监督微调

如图 3 所示，小语言模型（SLM）经历三阶段监督微调（SFT）过程，分别为热启动、下一步生成和展开。

在阶段 1（热启动）中，输入指令仅限于根节点所包含的信息，包括 NoSQL 数据库 D、对话历史 H_i以及当前问题 q_i。对应的目标响应为真实值 NoSQL 查询 y_i。此阶段旨在让小语言模型熟悉 NoSQL 查询的语法和语义。在阶段 2（下一步生成）中，输入指令会补充根节点信息以及此前生成的所有步骤。此阶段的目标响应是单个下一步，由一个可执行的 NoSQL 阶段及其配套的自然语言注释组成。此阶段旨在提升小语言模型遵循指令指导逐步生成的能力。在阶段 3（展开）中，输入指令会补充根节点信息以及此前生成的步骤序列，旨在使小语言模型能够一次性生成所有剩余步骤以及最终的 NoSQL 查询。这也构成了此阶段的目标响应。监督微调的核心目标是最小化训练集上的负对数似然损失（$\mathrm{L}_{\text{监督微调}}$），其公式如下：

$$\mathcal{L}_{\mathrm{SFT}} = -\frac{1}{L}\sum_{t=1}^{L}\log P\left(y_t \mid x, y_{<t}; \theta\right), \tag{2}$$

其中， L 表示目标响应的标记长度， y表示输入指令， x 表示语言模型的可学习参数。

3.4 自训练流程

鉴于小型语言模型的推理能力有限，我们采用多轮自训练框架，迭代生成高质量数据，并通过丰富多样的推理路径来扩充训练集。具体而言，自训练过程包含两个关键步骤。第一步是阶段增强的思维链生成，利用小语言模型采样多种推理路径。随后将这些新采样的路径纳入原始训练集；为降低训练复杂度和计算成本，更新后的训练集仅保留最近两轮生成的数据。此外，还会移除重复的推理数据。第二步是三阶段监督微调，利用经过优化的训练集对

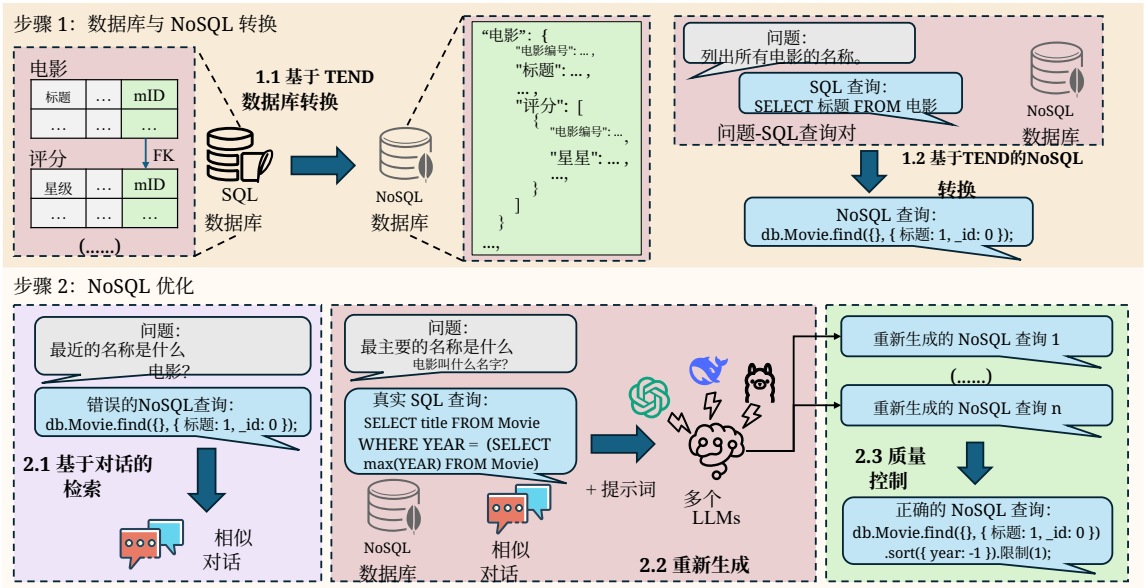


图 5：数据集构建流水线概览。关系数据库中的数据会转换为兼容 NoSQL 的数据。转换错误的数据会通过基于对话的检索增强生成方法进行优化。为确保数据质量，实施了包括人工审查在内的质量控制措施。

小语言模型进行自适应微调。随后，在下一轮中部署更新后的小语言模型，以生成质量更高的训练数据。完整细节请参见附录 A.2.2 节。

3.5 基于MCTS的测试时扩展

经过多轮自训练后，更新后的SLM被用于预测NoSQL 查询。为进一步提升最终答案的可靠性，采用第3.2节中的MCTS算法来扩展测试时计算。然而，在推理阶段无法获得真实查询。为实施奖励引导搜索策略，采用了自监督奖励模型，该模型已在众多研究中被证明是有效的 [27, 41]。奖励计算如下：

$$R(y, q_i, \mathcal{H}_i, \mathcal{D}) = \frac{1}{N}\sum_{j=1}^{N} \mathbb{1}\left[\text{Execute}(y, \mathcal{D}) = \text{Execute}\left(y_j, \mathcal{D}\right)\right] \tag{3}$$

其中，$\mathbb{1}$ 表示指示函数，Execute是用于在给定数据库上执行查询的运算符，y_j 表示采样得到的NoSQL 查询，N 表示样本数量。该奖励模型优先选择执行结果（即EVM）始终稳定的NoSQL 查询，使MCTS能够专注于可靠的推理轨迹，而无需标注数据。对于选择模型，采用贪心选择策略来确定最优NoSQL 查询。具体而言，在MCTS算法生成一组多样化的候选推理路径后，选择与奖励分数最高的路径相对应的NoSQL 查询。

3.6 框架验证

3.6.1 数据集构建。 图5展示了数据集构建的概况。首先，采用该研究中介绍的方法 TEND [36], 将关系数据库转换为NoSQL数据库，并将SQL查询转换为NoSQL 查询。随后，提出一种由多个大语言模型（LLM）支持的检索增强生成（RAG） [12, 26, 82] 方法，以优化转换后的NoSQL 查询。最后，通过人工审核等质量控制策略提升数据质量，确保其达到预期标准。

在此场景中，MongoDB [50, 60] 被用作 NoSQL 数据库的代表，这与现有的文本到 NoSQL [36, 46]研究一致。具体而言，借助基于 TEND 的方法 [36]，CoSQL [75] 中的关系数据库被转换为 NoSQL 数据库，所有 SQL 查询也都被转换为 NoSQL 查询。转换后的 NoSQL 数据库和 NoSQL 查询构成了本文提出的数据集 CoNoSQL。尤其是，关系数据库中表之间的外键关系被转换为 NoSQL 数据库中的嵌套关系。CoNoSQL 中移除了 2 个有问题的数据库及其关联的 24 段对话。为验证转换后的 NoSQL 查询是否正确，分别在 SQLite 数据库和 MongoDB 数据库上执行 SQL 查询和转换后的 NoSQL 查询。如果两者的结果相同（即 EVM=1），则认为转换后的 NoSQL 查询正确。

然而，通过流水线转换的 NoSQL 查询中，只有 60.13% 是正确的。为了进一步提高正确 NoSQL 查询的比例，我们提出了一种由多个大语言模型支持的检索增强生成方法。具体而言，根据转换后的 NoSQL 查询是否正确，将 CoNoSQL 拆分为两个不同的部分。一个子集，即参考集，包含已移除所有错误问题-查询对的对话。另一个集合则包含所有待优化的错误问题-查询对。例如，给定一个错误的 NoSQL 查询“db.Movie.find({}, { title: 1,_id: 0 })”及其对应的用户问题“最近上映的电影叫什么名字”，

流水线会通过计算问题嵌入向量之间的距离，在参考集中搜索相关的问题-查询对。随后，会检索与这些相关对相对应的对话，并将其作为相似对话提供。接着，向多个大语言模型提供一条提示词，其中包含用户问题、错误的 NoSQL 查询、NoSQL 数据库信息和相似对话。该提示词见附录 B 节中的图 12。此方法旨在使大语言模型能够更准确地重新生成 NoSQL 查询。经过这一过程，正确 NoSQL 查询的比例提升至 89.61%。

3.6.2 数据集统计与分析。本节讨论 CoNoSQL 数据集的统计信息。表 1 显示，在跨域设置下，CoNoSQL 数据集被划分为训练集和测试集。每段对话的平均轮次数超过 4，体现了 CoNoSQL 数据集的语义复杂性。详细的统计信息与分析见附录 B。

表 1：数据集统计

	训练集	Test	总体
数据库总数	139	19	158
集合总数	335	37	372
对话总数	2,115	276	2,391
总轮数	8,484	1,224	9,708
平均每段对话轮数	4.01	4.43	4.06
平均每轮问题词元数	11.91	11.79	11.90
平均每轮 NoSQL 词元数	31.38	32.98	31.58
Find 方法总数	1,723	291	2,014
Aggregate 方法总数	6,757	926	7,683
其他方法总数	4	7	11

4 实验

4.1 设置

4.1.1 数据集。我们使用 CoNoSQL 评估所提出的方法和基线方法的性能。为确保数据集质量，我们进行了后处理：从训练集中移除执行不匹配查询；测试集则通过人工干预进行人工验证（指标包括执行准确率和语义等价性） [6, 62]。详细信息见附录 B.3 节。

4.1.2 基础模型与设置。我们采用先进水平的大语言模型构建基线方法，以评估 CoNoSQL 数据集，包括 DeepSeek-R1 [16]、Qwen2.5-Coder-7B-Instruct [19], 等开源代表模型，以及闭源模型 GPT-4o [20]。由于 GPU资源有限，我们仅针对所提出方法中的 Qwen2.5-Coder-7B-Instruct 进行了三轮自我进化。详细的实现步骤和提示模板见附录 C 节。

4.1.3 基线方法。 在评估中，我们采用了若干免训练范式，即零样本学习 [34, 48], 、少样本学习和检索增强生成（RAG） [12, 26, 82]。少样本学习和 RAG 方法均采用十样本设置。为了构建稳健的监督微调（SFT）基线，我们实现了两种方法：直接NoSQL预测和自举式 STaR-SFT 方法 [44, 78]。具体而言，STaR-SFT 使用 DeepSeek-R1 生成的分治式思维链推理数据进行训练。此外，我们纳入了 SMART [36], , 这是一种具有代表性的文本到 NoSQL方法。我们还报告了成熟的基于智能体的方法的性能，包括 ReAct [71] 和 Plan-and-Solve [61]。详细的实现步骤和提示词模板见附录 C 节。

4.1.4 指标。本实验采用了其他文本到 NoSQL 任务中使用的指标 [36,46] , 包括精确匹配（*EM*）和执行准确率（*EX*）。*EM*衍生出两个指标：查询阶段匹配（*QSM*）和查询字段覆盖率（*QFC*）。*EX*也包含两个细分指标：执行字段匹配（*EFM*）和执行值匹配（*EVM*）。在评估重点方面，EM、QSM 和 QFC 用于评估生成查询的语法正确性，而 EX、EFM 和 EVM 则用于验证其语义正确性。鉴于 NoSQL 数据库具有灵活性，字段名称完全可以由用户自定义，因此我们在分析中优先采用 EVM，而非 EX，以减轻自定义字段命名约定造成的干扰。所有指标的定义均见附录 C.2 节。

4.2 主要实验结果

4.2.1 对比。主要结果见表2。与其他基线模型相比，Stage-MCTS 在 EVM 准确率方面表现更优，而该指标是衡量 NoSQL 生成任务语义准确性的有效指标。值得注意的是，与零样本大规模模型和基于检索增强生成的小样本模型相比，我们的方法在 EVM 准确率上分别取得了 6.7% 和 1.39% 的相对提升。该结果表明，参数量为 70 亿的小型语言模型能够应对具有挑战性的 NoSQL 生成任务，性能可与 GPT-4o 和 DeepSeek-R1 等规模大得多的模型相当。

关于基于SFT的方法，我们的方法无论是否采用测试时缩放策略（分别见表2和表 3），都表现更优，这验证了所提出的阶段增强的思维链推理数据的有效性。

然而，基于智能体的方法在所提出的任务上表现不佳。一个可能的原因是这类方法本身缺乏推理能力，这使它们在面对未见过的 NoSQL 数据库时难以构建有效查询。

作为对我们方法的扩展，我们采用强大的推理模型 DeepSeek-R1 作为骨干模型，在第1轮生成推理数据。这一策略加速了自训练，也与先前研究的发现一致 [14]。在后续所有自训练轮次中，我们部署小型语言模型以降低计算预算。该方法的表现优于先进水平的大型推理模型，在零样本设置下将EVM准确率提高了最多 7.93%。

4.2.2 消融研究。 我们进行了消融研究，以探究各个模块的作用。主要结果见表 3和表4，附录C.3节还提供了其他消融结果。

各模块的评估。我们分别验证两个核心组件的有效性：基于奖励的采样与路径优化（Reward-PR）组件，以及基于MCTS的测试时扩展（MCTS-TTS）组件。所有结果均来自

表2：不同方法在六项指标上的实验结果。粗体值表示所有方法中的最佳性能，下划线值表示基线方法中的最佳性能。

	方法	模型规模	基于查询 EM	QSM	QFC	基于执行 EX	EFM	EVM
Training-free Paradigms	零样本（GPT-4o）	UNK	0.1413	0.4649	0.6242	0.4248	0.5433	0.5654
	零样本（DeepSeek-R1）	671B	0.1144	0.4575	0.6495	0.4779	0.5817	0.6168
	少样本（GPT-4o）	UNK	**0.1716**	0.5155	0.6667	0.4967	0.6078	0.6577
	少样本（DeepSeek-R1）	671B	0.1601	0.5033	0.6634	0.5106	0.6176	0.6593
	检索增强生成（GPT-4o）	UNK	0.1699	**0.5261**	**0.6683**	0.5033	0.6152	0.6503
	检索增强生成 (DeepSeek-R1)	671B	0.1422	0.5155	0.6618	0.5270	**0.6299**	0.6699
SFT	直接监督微调 (Qwen2.5-Coder)	7B	0.1511	0.5008	0.5735	0.4730	0.5572	0.5899
	STaR-SFT [78] (Qwen2.5-Coder, DeepSeek-R1)	7B	0.1021	0.4461	0.6373	0.4649	0.5531	0.6454
Agent	ReAct [71](GPT-4o)	UNK	0.0891	0.3954	0.5605	0.3587	0.4706	0.4559
	Plan-and-Solve [61] (GPT-4o)	UNK	0.0760	0.3734	0.6038	0.4175	0.4918	0.5417
	SMART [36] (GPT-4o)	UNK	0.1111	0.4902	0.5931	0.4583	0.5400	0.5654
Ours	Stage-MCTS（Qwen2.5-Coder）	7B	0.1462	0.5221	0.6552	**0.5392**	0.6266	0.6838
	Stage-MCTS（Qwen2.5-Coder，DeepSeek-R1）	7B	0.1332	0.5049	0.6528	0.5335	0.6144	**0.6961**

三轮自我进化。对于消融了 Reward-PR 组件的情况，我们采用拒绝采样策略评估模型性能，仅保留答案正确的 NoSQL 查询所对应的推理路径作为训练集。对于消融了 MCTS-TTS 组件的情况，我们将输出的推理路径数量设为 1。表 3 显示，消融任一组件后，模型在所有指标上的性能均有所下降。这一结果验证了每个模块都对整体性能有积极贡献。

表 3：经过三轮自训练后对各模块进行的消融研究。

方法	基于查询 EM	QSM	QFC	基于执行 EX	EFM	EVM
StageMCTS	0.1462	0.5221	0.6552	0.5392	0.6266	0.6838
不使用 Reward-PR	0.0629	0.3930	0.5792	0.4526	0.5368	0.6176
不使用 MCTS-TTS	0.1413	0.5123	0.6479	0.5180	0.6225	0.6601

自训练流程评估。鉴于小型语言模型的能力有限，我们实施了三轮基于MCTS的深入
推理以迭代生成更高质量的数据，并用更具挑战性的 NoSQL 生成样本扩充训练集。研究的主要发现见表4。未经过任何微调的基础模型记为“SLM-base”，其经过第1轮、第2轮和第3轮微调后的变体分别标记为“SLM-r1”、“SLM-r2”和“SLM-r3”。具体而言，我们在研究中采用 Qwen2.5-Coder-7B-Instruct 作为代表性小语言模型。这项消融研究表明，定制的自训练流水线能够有效推动性能持续提升。此外，表中结果显示，随着训练轮数增加，各项指标的增长率逐渐降低，表明模型性能正趋于收敛。

4.2.3 案例研究。 在我们的实验设置下，蒙特卡洛树搜索的搜索树包含 50 多个节点，总计有 10 多条不同的推理路径。由于篇幅限制，我们仅在本案例研究中展示一个具有代表性的分支作为示例。如图6所示，所提出的方法能够生成

表4：自训练小语言模型在每一轮中的性能，显示出持续改进。

轮次#	基于查询 EM	QSM	QFC	基于执行 EX	EFM	EVM
SLM-base	0.0882	0.4232	0.5580	0.3905	0.4771	0.5008
SLM-r1	0.1250	0.4935	0.6520	0.5270	0.6201	0.6675
SLM-r2	0.1397	0.5106	0.6503	**0.5433**	**0.6266**	0.6806
SLM-r3	**0.1462**	**0.5221**	**0.6552**	0.5392	**0.6266**	**0.6838**

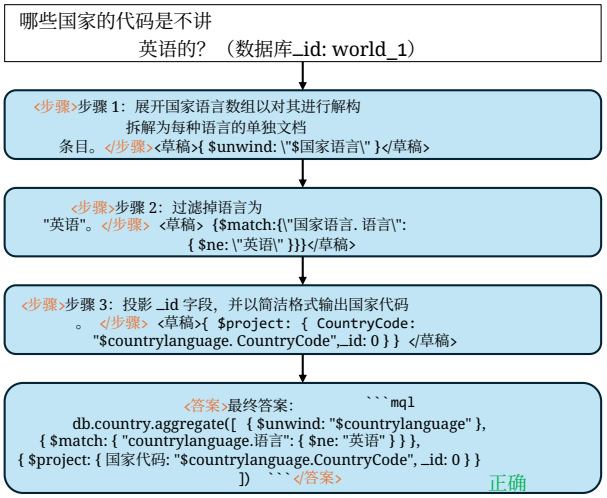


图6：MCTS搜索树中的一个示例分支（因页面限制省略了部分节点）。

依次执行各阶段及其对应的注释。值得注意的是，它隐式遵循计划与求解范式，其中每个推理步骤及其生成的草稿阶段都建立在先前步骤的基础上。有关更全面的结果，请参阅附录中的 C.4 节。这一补充结果表明，在给定相同先前步骤的情况下，我们的方法可以枚举多个候选阶段，随后基于奖励模型执行引导式搜索。

4.3 总结

Stage-MCTS 在所提出的任务上实现了更高的 EVM准确率，较零样本大语言模型最高提升 7。93%，较基于检索增强生成的少样本模型最高提升 2。62%。结果表明，70亿参数小语言模型能够达到更大规模模型（如 GPT-4o）的性能。消融研究证实，Reward-PR 和 MCTS-TTS 两个组件都至关重要，而自训练流程在多轮迭代中带来了持续改进。

5 相关工作

本研究与用户友好型 NoSQL 数据库、文本到 SQL 和大语言模型推理领域密切相关。

用户友好型 NoSQL 数据库。

不仅是SQL（NoSQL）数据库是先进的数据存储与管理系统，能够以非表格格式存储数据，且无需预定义结构。在数据库和数据挖掘领域，现有关于NoSQL数据库的研究主要聚焦于几个关键领域。例如，研究 [3] 探讨了NoSQL数据库系统的可用性和可扩展性。另一篇论文 [39] 考察了NoSQL数据库中的安全性和隐私问题。此外，研究 [37]旨在通过查询优化提高NoSQL数据库的能源效率。还有研究 [1] 致力于在云计算环境中实现数据库管理系统的可扩展性、弹性和自治性。

尽管各领域已开展大量NoSQL研究，现有工作却忽视了缺乏NoSQL查询知识的非专业用户的易用性问题。为此，有研究 [81] 提出了面向医疗保健的文本转ESQ（Elasticsearch 查询）任务，使用户能够在NoSQL环境中精准、高效地探索复杂的医疗保健数据。此外，研究 [80] 提出了一种可解释的思维链方法，利用大语言模型完成文本转ESQ任务。近期，研究 [36]提出了文本到NoSQL任务，支持通过自然语言与NoSQL数据库交互，并构建了TEND数据集（用于MongoDB查询），以支持跨领域划分。随后，研究 [46] 推出了Multi-TEND，这是面向多语言文本到 NoSQL任务的最大多语言基准。

然而，先前的工作 [36, 46, 80, 81] 侧重于将单个问题转换为对应的 NoSQL 查询，而对话式文本到 NoSQL 任务旨在将依赖上下文的问题映射到相应的 NoSQL 查询。

文本到 SQL。 文本到 SQL 对降低访问关系数据库 [18, 22, 35] [28, 40, 的 53]

门槛具有重要意义。早期的工作主要依赖基于规则的方法。这些方法基于精心设计的规则和模板实现，尤其适用于简单场景。随后，深度神经网络 [5, 66, 74, 84] 和预训练语言模型（PLMs） [8, 47] 被开发出来，用于文本到 SQL 任务，并取得了令人鼓舞的成果。具体而言，TaBERT [72] 在大量表格数据和英语语境语料上进行训练，以联合理解文本与表格数据。RESDSQL [29] 被设计为一种融合排序增强编码与骨架感知解码的框架，通过解耦模式链接和骨架解析，提高了准确性和稳健性。Graphix-T5 [31] 通过引入专门设计的图感知层，增强了预训练的 T5 模型 [47]，旨在提升模型的推理能力。

为了评估文本到 SQL 模型在实际场景中的运行效果，人们开发并发布了多个大型基准数据集，包括 WikiSQL [84],、Spider [76],、SParC [77],、CoSQL [75],、BIRD [32] 和 Spider 2.0 [25]。

近年来，人们对大语言模型（LLM）方法的兴趣日益浓厚。基于大语言模型的方法通过上下文学习 [7, 11, 42, 49, 52, 56, 59] 和监督微调（SFT） [13, 30, 43, 45] 范式实现了文本到 SQL，并凭借精心设计的框架和更强的理解能力，取得了先进水平的准确性。

然而，所提出的对话式文本到 NoSQL 任务与文本到 SQL 任务有所不同，尤其是 SParC 和 CoSQL。例如，对话式文本到 NoSQL 旨在从集合中检索数据，而文本到 SQL 则侧重于表。处理复杂问题时，文本到 SQL 必须生成能够跨不同关系表执行联合搜索的查询。相比之下，对话式文本到 NoSQL 通过查询包含嵌套字段的单个文档来解决这一问题。此外，NoSQL 数据库的模式比 SQL 数据库的模式更加灵活，因此对话式文本到 NoSQL 任务也更复杂。

大语言模型推理。 借助思维链（CoT）技术 [24, 64], ，大语言模型在广泛的任务中取得了显著的性能提升 [14, 17, 54] 。

作为一种具有代表性的推理范式，CoT 模拟人类的认知过程，将复杂问题分解为一系列推理步骤，从而得出连贯的解决方案。在此基础上，思维树（ToT）框架 [70]进一步拓展了 CoT 方法。该框架构建了树状推理空间，其中包含多条备选推理路径，并探索语义连贯的思维单元，以实现审慎而周全的决策。

现有关于大语言模型推理的研究主要聚焦于合成大规模、多样化的数据集 [10, 55, 63, 73]，以及设计更有效的算法，以增强大语言模型的推理能力 [57, 69, 78, 79]。然而，大多数研究都集中在数学推理和科学任务上。为弥补这一空白，本文首次尝试利用一种改进的、基于MCTS的数据增强方法和量身定制的自训练流程，提升大语言模型在数据库领域的推理能力。

6 结论

在本研究中，我们识别了单轮文本到 NoSQL 系统的局限性，并提出了新颖的对话文本到 NoSQL 任务，以实现多轮且具备上下文感知的查询生成。我们引入了 Stage-MCTS，该框架将 NoSQL 查询生成建模为搜索问题，并通过基于MCTS的数据增强、渐进式微调和迭代自训练，为小型语言模型配备 NoSQL 特定推理能力。为支持本研究，我们构建了 CoNoSQL，这是一个通过大语言模型与人类协作构建的大规模跨领域数据集。大量实验验证了我们方法的有效性，其在 EVM准确率上优于 DeepSeek-R1 和 GPT-4o 等先进模型。

未来的工作将重点关注将对话式查询系统集成到更广泛的数据探索工作流中，包括自动化可视化生成和深度研究智能体。

参考文献

[1] Divyakant Agrawal、Amr El Abbadi、Sudipto Das 和 Aaron J Elmore。2011年。云中的数据库可扩展性、弹性和自治性。载于高级应用数据库系统国际会议。Springer，2–15。[2] Elena Baralis、Andrea Dalla Valle、Paolo Garza、Claudio Rossi 和 Francesco Scullino。2017年。用于地理空间应用的 SQL 与 NoSQL 数据库。载于2017年 IEEE国际大数据会议（Big Data）。IEEE，3388–3397。[3] Rick Cattell。2011年 39，4 2011年 12 27。可扩展的 SQL 和 NoSQL 数据存储。ACM SIGMOD Record（），–。[4] Dong Chen、Shaoxin Lin、Muhan Zeng、Daoguang Zan、Jian-Gang Wang、Anton Cheshkov、Jun Sun、Hao Yu、Guoliang Dong、Artem Aliev 等。2024年。Coder：利用多智能体和任务图解决问题。arXiv预印本 arXiv:2406.01304（2024年）。[5] DongHyun Choi、Myeong Cheol Shin、EungGyun Kim 和 Dong Ryeol Shin。2021年。Ryansql：通过递归应用基于草图的槽位填充，实现跨领域数据库中的复杂文本到 SQL。计算语言学 47，2（2021年），309–332。[6] John Chung、Ece Kamar 和 Saleema Amershi。2023年。在保持准确性的同时提高多样性：使用大语言模型和人工干预生成文本数据。载于计算语言学协会年会第61届会议论文集（第1卷：长论文）。575–593。[7] Minghang Deng、Ashwin Ramachandran、Canwen Xu、Lanxiang Hu、Zhewei Yao、Anupam Datta 和 Hao Zhang。2025年。ReFoRCE：具备自我优化、格式限制和列探索功能的文本转 SQL 智能体。载于 ICLR 2025研讨会：VerifAI：现实世界中的人工智能验证。https://openreview.net/forum?id=OuFIfDBwQd [8] Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2019年。BERT：用于语言理解的深度双向变换器预训练。载于2019年计算语言学协会北美分会会议：人类语言技术论文集，第1卷（长论文和短论文）。4171–4186。[9] Abhimanyu Dubey、Abhinav Jauhri、Abhinav Pandey、Abhishek Kadian、Ahmad Al-Dahle、Aiesha Letman、Akhil Mathur、Alan Schelten、Amy Yang、Angela Fan 等。2024年。Llama 3 模型群。arXiv预印本 arXiv:2407.21783（2024年）。[10] Bofei Gao、Feifan Song、Zhe Yang、Zefan Cai、Yibo Miao、Qingxiu Dong、Lei Li、Chenghao Ma、Liang Chen、Runxin Xu、Zhengyang Tang、Benyou Wang、Daoguang Zan、Shanghaoran Quan、Ge Zhang、Lei Sha、Yichang Zhang、Xuancheng Ren、Tianyu Liu 和 Baobao Chang。2025年。Omni-MATH：面向大语言模型的通用奥林匹克级数学基准。载于第十三届国际学习表征会议。https: // openreview.net/forum?id=yaqPf0KAlN[11] Dawei Gao、Haibin Wang、Yaliang Li、Xiuyu Sun、Yichen Qian、Bolin Ding 和 Jingren Zhou。2024年。大语言模型赋能的文本转 SQL：基准评估。VLDB基金会会刊 17，5（2024年），1132–1145。[12] Yunfan Gao、Yun Xiong、Xinyu Gao、Kangxiang Jia、Jinliu Pan、Yuxi Bi、Yi Dai、Jiawei Sun、Qianyu Guo、Meng Wang 等。2023年。面向大语言模型的检索增强生成：综述。CoRR（2023年）。[13] Satya Krishna Gorti、Ilan Gofman、Zhaoyan Liu、Jiapeng Wu、Noël Vouitsis、Guangwei Yu、Jesse C. Cresswell 和 Rasa Hosseinzadeh。2025年。MSc-SQL：用于文本转 SQL 翻译的多样本批评小语言模型。载于2025年 ACL美洲分会年会。https: //arxiv.org/abs/2410.12916[14] Xinyu Guan、Li Lyna Zhang、Yifei Liu、Ning Shang、Youran Sun、Yi Zhu、Fan Yang 和 Mao Yang。2025年。rStar-Math：小型大语言模型通过自我进化的深度思考掌握数学推理。载于第四十二届国际机器学习会议。https://openreview.net/forum?id=5 zwF1GizFa [15] Dongming Guo 和 Erling Onstein。2020年。NoSQL 数据库中的先进水平地理空间信息处理。ISPRS国际地理信息期刊 9，5 2020 331（），。[16] Daya Guo、Dejian Yang、Haowei Zhang、Junxiao Song、Ruoyu Zhang、Runxin Xu、Qihao Zhu、Shirong Ma、Peiyi Wang、Xiao Bi 等。2025年。DeepSeek-R1：通过强化学习激发大语言模型的推理能力。arXiv预印本 arXiv:2501.12948（2025年）。[17] Shibo Hao、Yi Gu、Haodi Ma、Joshua Hong、Zhen Wang、Daisy Wang 和 Zhiting Hu。2023年。使用语言模型进行推理就是利用世界模型进行规划。载于2023年自然语言处理实证方法会议论文集。8154–8173。[18] Zijin Hong、Zheng Yuan、Qinggang Zhang、Hao Chen、Junnan Dong、Feiran Huang 和 Xiao Huang。2025年。新一代数据库接口：基于大语言模型的文本转 SQL 综述。IEEE 知识与数据工程汇刊 37，12（2025年），7328–7345。doi:10.1109/TKDE.2025.3609486[19] Binyuan Hui、Jian Yang、Zeyu Cui、Jiaxi Yang、Dayiheng Liu、Lei Zhang、Tianyu Liu、Jiajun Zhang、Bowen Yu、Keming Lu 等。2024年。Qwen2. 5-coder 技术报告。arXiv预印本 arXiv:2409.12186（2024年）。[20] Aaron Hurst、Adam Lerer、Adam P Goucher、Adam Perelman、Aditya Ramesh、Aidan Clark、AJ Ostrow、Akila Welihinda、Alan Hayes、Alec Radford 等。2024年。GPT-4o 系统卡。arXiv预印本 arXiv:2410.21276（2024年）。

[21] Sultana Kalid、Ali Syed、Azeem Mohammad 和 Malka N Halgamuge。2017年。大数据 NoSQL 数据库："Big-Table""DynamoDB"和"Cassandra"的比较与分析。载于 2017年 IEEE 第二届国际大数据分析会议（ICBDA）。IEEE，89–93。[22] George Katsogiannis-Meimarakis 和 Georgia Koutrika。2023年。深度学习文本到 SQL 方法综述。VLDB 期刊 32，4（2023年），905–936。[23] Levente Kocsis 和 Csaba Szepesvári。2006年。基于赌博机的蒙特卡洛规划。载于欧洲机器学习会议。Springer，282–293。[24] Takeshi Kojima、Shixiang Shane Gu、Machel Reid、Yutaka Matsuo 和 Yusuke Iwasawa。2022年。大语言模型是零样本推理器。神经信息处理系统进展 35（2022年），22199–22213。[25] Fangyu Lei、Jixuan Chen、Yuxiao Ye、Ruisheng Cao、Dongchan Shin、Hongjin SU、ZHAOQING SUO、Hongcheng Gao、Wenjing Hu、Pengcheng Yin、Victor Zhong、Caiming Xiong、Ruoxi Sun、Qian Liu、Sida Wang 和 Tao Yu。2025年。Spider 2.0：评测语言模型在真实世界企业文本到 SQL 工作流中的表现。载于第十三届国际学习表征会议。https://openreview.net/forum?id=XmProj9cPs[26] Patrick Lewis、Ethan Perez、Aleksandra Piktus、Fabio Petroni、Vladimir Karpukhin、Naman Goyal、Heinrich Küttler、Mike Lewis、Wen-tau Yih、Tim Rocktäschel 等。2020年。面向知识密集型 NLP 任务的检索增强生成。神经信息处理系统进展 33（2020年），9459–9474。[27] Boyan Li、Jiayi Zhang、Ju Fan、Yanwei Xu、Chong Chen、Nan Tang 和 Yuyu Luo。2025年。Alpha-SQL：使用蒙特卡洛树搜索实现零样本文本到 SQL。载于第四十二届国际机器学习会议。https://openreview. net/forum?id=kGg1ndttmI [28] Fei Li 和 Hosagrahar V Jagadish。2014年。为关系数据库构建交互式自然语言接口。VLDB基金会会刊 8，1（2014年），73–84。[29] Haoyang Li、Jing Zhang、Cuiping Li 和 Hong Chen。2023年。RESDSQL：解耦文本到 SQL 的模式链接与骨架解析。载于 AAAI人工智能会议论文集，第37卷。13067–13075。[30] Haoyang Li、Jing Zhang、Hanbing Liu、Ju Fan、Xiaokang Zhang、Jun Zhu、Renjie Wei、Hongyan Pan、Cuiping Li 和 Hong Chen。2024年。CodeS：致力于构建用于文本到 SQL 的开源语言模型。ACM数据管理会刊 2，3，文章127（2024年5月），28页。doi:10.1145/3654930[31] Jinyang Li、Binyuan Hui、Reynold Cheng、Bowen Qin、Chenhao Ma、Nan Huo、Fei Huang、Wenyu Du、Luo Si 和 Yongbin Li。2023年。Graphix-T5：将预训练 Transformer 与图感知层相结合，用于文本到 SQL 解析。载于 AAAI 人工智能会议论文集，第 37 卷。13076–13084。[32] Jinyang Li 、 Binyuan Hui、Ge Qu、Jiaxi Yang、Binhua Li、Bowen Li、Bailin Wang、Bowen Qin、Ruiying Geng、Nan Huo 等。2023年。大语言模型能否胜任数据库接口？一个面向大规模数据库落地文本到 SQL 的基准。神经信息处理系统进展 36（2023年），42330–42357。[33] Hunter Lightman、Vineet Kosaraju、Yuri Burda、Harrison Edwards、Bowen Baker、Teddy Lee、Jan Leike、John Schulman、Ilya Sutskever 和 Karl Cobbe。2024年。让我们逐步验证。载于第十二届国际学习表征会议。https://openreview.net/forum?id=v8L0pN6EOi [34] Aiwei Liu、Xuming Hu、Lijie Wen 和 Philip S Yu。2023年。全面评估 ChatGPT 的零样本文本到 SQL 能力。arXiv 预印本 arXiv:2303.13547（2023年）。[35] Xinyu Liu、Shuyu Shen、Boyan Li、Peixian Ma、Runzhi Jiang、Yuxin Zhang、Ju Fan、Guoliang Li、Nan Tang 和 Yuyu Luo。2025年。大语言模型时代的文本到 SQL 综述：现状与未来方向？IEEE 知识与数据工程汇刊

37, 10 2025 5735 5754 10 1109 2025 3592032 ( ), – 。[36] doi: 。/TKDE. 。Jinwei Lu、Yuanfeng Song、Zhiqian Qin、Haodi Zhang、Chen Zhang 和 Ray-mond Chi-Wing Wong。2025年。弥合差距：通过文本到 NoSQL 翻译实现 NoSQL 数据库的自然语言查询。arXiv 预印本 arXiv:2502.11201（2025年）。[37] Divya Mahajan、Cody Blakeney 和 Ziliang Zong。2019年。通过查询优化提升关系数据库和 NoSQL 数据库的能源效率。可持续计算：信息学与系统 22（2019年），120–133。[38] ABM Moniruzzaman 和 Syed Akhter Hossain。2013年。NoSQL 数据库：大数据分析新时代数据库——分类、特征与比较。《国际数据库理论与应用期刊》6，4（2013年）。[39] Lior Okman、Nurit Gal-Oz、Yaron Gonen、Ehud Gudes 和 Jenny Abramov。2011年。NoSQL 数据库中的安全问题。载于 2011年 IEEE 第十届计算与通信中的信任、安全与隐私国际会议。IEEE，541–547。[40] Ana-Maria Popescu、Alex Armanasu、Oren Etzioni、David Ko 和 Alexander Yates。2004年。现代数据库自然语言接口：将统计解析与语义可处理性相结合。载于 COLING 2004：第二十届国际计算语言学会议论文集。141–147。[41] Mohammadreza Pourreza、Hailong Li、Ruoxi Sun、Yeounoh Chung、Shayan Talaei、Gaurav Tarlok Kakkar、Yu Gan、Amin Saberi、Fatma Ozcan 和 Sercan O Arik。2025年。CHASE-SQL：文本到 SQL 中的多路径推理与偏好优化候选项选择。载于第十三届国际学习表征会议。https://openreview.net/forum?id=CvGqMD5OtX

[42] Mohammadreza Pourreza 和 Davood Rafiei。2023年。Din-sql：通过自我纠正实现文本到 SQL 的上下文分解式学习。神经信息处理系统进展 36（2023年），36339–36348。[43] Mohammadreza Pourreza 和 Davood Rafiei。2024年。DTS-SQL：使用小型大语言模型进行分解式文本到 SQL。载于计算语言学协会论文集：EMNLP 2024 论文集。8212–8220。[44] Mohammadreza Pourreza、Shayan Talaei、Ruoxi Sun、Xingchen Wan、Hailong Li、Azalia Mirhoseini、Amin Saberi 和 Sercan O Arik。2025年。Reasoning-SQL：通过针对 SQL 定制的部分奖励进行强化学习，以增强文本到 SQL 推理。载于第二届语言建模会议。https: //openreview.net/forum?id=HbwkIDWQgN[45] Yang Qin、Chao Chen、Zhihang Fu、Ze Chen、Dezhong Peng、Peng Hu 和 Jieping Ye。2025年。ROUTE：面向文本到 SQL 的鲁棒多任务调优与协作。载于第十三届国际学习表征会议。https://openreview.net/forum?id = 6 0 BAglD NGy Zhiqian Qin、[46] Yuanfeng Song、Jinwei Lu、Yuanwei Song、Shuaimin Li 和 Chen Ja-son Zhang。2025年。MultiTEND：自然语言到 NoSQL 查询翻译的多语言基准。载于计算语言学协会论文集：ACL 2025 论文集，Wanxiang Che、Joyce Nabende、Ekaterina Shutova 和 Mohammad Taher Pilehvar（编）。计算语言学协会，奥地利维也纳，24632–24657。doi:10.18653/v1/2025.findings-acl.1265[47] Colin Raffel、Noam Shazeer、Adam Roberts、Katherine Lee、Sharan Narang、Michael Matena、Yanqi Zhou、Wei Li 和 Peter J Liu。2020年。探索统一文本到文本转换器迁移学习的局限性。机器学习研究期刊 21，140（2020年），1–67。[48] Nitarshan Rajkumar、Raymond Li 和 Dzmitry Bahdanau。2022年。评估大语言模型的文本到 SQL 能力。arXiv 预印本 arXiv:2204.00498（2022年）。[49] Tonghui Ren、Yuankai Fan、Zhenying He、Ren Huang、Jiaqi Dai、Can Huang、Yinan Jing、Kai Zhang、Yifan Yang 和 X Sean Wang。2024年。Purple：让大语言模型成为更出色的 SQL编写器。载于 2024年 IEEE 第四十届国际数据工程会议（ICDE）。IEEE，15–28。[50] William Schultz、Tess Avitabile 和 Alyson Cabral。2019年。MongoDB 中可调的一致性。VLDB基金会会刊 12，12（2019年），2071–2081。[51] Poly Sil Sen 和 Nandini Mukherjee。2024年。基于本体的方法：为半结构化和非结构化健康数据设计 NoSQL 数据库。集群计算 27，1 （ 2024 年），959–976。[52] Jie Shi 、 Bo Xu 、 Jiaqing Liang、Yanghua Xiao、Jia Chen、Chenhao Xie、Peng Wang 和 Wei Wang。2025年。Gen-SQL：通过伪模式连接自然语言问题与数据库模式，实现高效文本到 SQL。载于第三十一届国际计算语言学会议论文集。3794–3807。[53] Niculae Stratica、Leila Kosseim 和 Bipin C Desai。2005年。使用语义模板构建 CINDI 虚拟图书馆的自然语言接口。数据与知识工程 55，1（2005年），4–19。[54] DiJia Su、Sainbayar Sukhbaatar、Michael Rabbat、Yuandong Tian 和 Qinqing Zheng。2025年。Dualformer：通过随机化推理轨迹学习实现可控的快思考与慢思考。载于第十三届国际学习表征会议。https://openreview.net/forum?id=bmbRCRiNDu[55] Liangtai Sun、Yang Han、Zihan Zhao、Da Ma、Zhennan Shen、Baocai Chen、Lu Chen 和 Kai Yu。2024年。Scieval：面向科学研究的多层次大语言模型评测基准。载于 AAAI人工智能会议论文集，第38卷。19053–19061。[56] Shayan Talaei、Mohammadreza Pourreza、Yu-Chen Chang、Azalia Mirhoseini 和 Amin Saberi。2024年。Chess：用于高效 SQL合成的上下文利用。arXiv 预印本 arXiv:2405.16755（2024年）。[57] Kimi Team、Angang Du、Bofei Gao、Bowei Xing、Changjiu Jiang、Cheng Chen、Cheng Li、Chenjun Xiao、Chenzhuang Du、Chonghua Liao 等。2025年。Kimi k1.5：利用大语言模型扩展强化学习。arXiv 预印本 arXiv:2501.12599（2025年）。[58] Dimpal Tomar、Jai Prakash Bhati、Pradeep Tomar 和 Gurjit Kaur。2019年。将医疗关系数据库迁移至 NoSQL云数据库，以用于医疗分析与管理。载于《医疗数据分析与管理》。Elsevier，59–87。[59] Rishit Toteja、Arindam Sarkar 和 Prakash Mandayam Comar。2025年。结合检索增强生成的上下文强化学习，用于文本到 SQL。载于第三十一届国际计算语言学会议论文集。10390–10397。[60] Misha Tyulenev、Andy Schwerin、Asya Kamsky、Randolph Tan、Alyson Cabral 和 Jack Mulrow。2019年。在 MongoDB 中实现集群范围逻辑时钟和因果一致性。载于 2019年国际数据管理会议论文集。636–650。[61] Lei Wang、Wanyu Xu、Yihuai Lan、Zhiqiang Hu、Yunshi Lan、Roy Ka-Wei Lee 和 Ee-Peng Lim。2023年。计划与求解提示：利用大语言模型改进零样本思维链推理。载于计算语言学协会第六十一届年会论文集（第一卷：长篇论文）。2609–2634。[62] Shuohang Wang、Yang Liu、Yichong Xu、Chenguang Zhu 和 Michael Zeng。2021年。想降低标注成本？GPT-3 可以帮忙。载于计算语言学协会论文集：EMNLP 2021 论文集，Marie-Francine Moens、Xuanjing Huang、Lucia Specia 和 Scott Wen-tau Yih（编）。计算语言学协会 语言学，蓬塔卡纳，多米尼加共和国，4195–4205。doi:10.18653/v1/2021.findings-emnlp.[63] Xiaoxuan Wang、Ziniu Hu、Pan Lu、Yanqiao Zhu、Jieyu Zhang、Satyen Subra- maniam、Arjun R Loomba、Shichang Zhang、Yizhou Sun 和 Wei Wang。2024年。SciBench：评估大语言模型的大学水平科学问题求解能力。载于第四十一届国际机器学习大会。https://openreview.net/forum?id=bq1JEgioLr[64] Jason Wei、Xuezhi Wang、Dale Schuurmans、Maarten Bosma、Fei Xia、Ed Chi、Quoc V Le、Denny Zhou 等。2022年。思维链提示能够激发大语言模型的推理能力。神经信息处理系统进展 35 (2022年)，24824–24837。[65] Thomas Wolf、Lysandre Debut、Victor Sanh、Julien Chaumond、Clement Delangue、Anthony Moi、Pierric Cistac、Tim Rault、Rémi Louf、Morgan Funtowicz 等。2019年。Hugging Face 的 Transformers：先进水平的自然语言处理。*arXiv* 预印本 *arXiv:1910.03771* (2019年)。[66] Xiaojun Xu、Chang Liu 和 Dawn Song。2017年。SQLNet：无需强化学习，从自然语言生成结构化查询。*arXiv* 预印本 *arXiv:1711.04436* (2017年)。[67] Chao-Tung Yang、Jung-Chun Liu、Wen-Hung Hsu、Hsin-Wen Lu 和 William Cheng-Chung Chu。2013年。将数据转换方法应用于医疗保健数据的 NoSQL 数据库。载于*2013*年国际并行与分布式计算、应用及技术会议。IEEE，198–205。[68] John Yang、Carlos Jimenez、Alexander Wettig、Kilian Lieret、Shunyu Yao、Karthik Narasimhan 和 Ofir Press。2024年。SWE-agent：智能体—计算机接口助力自动化软件工程。神经信息处理系统进展 37 (2024年)，50528–50652。[69] Wenkai Yang、S huming Ma、Yankai Lin 和 Furu Wei。2025年。迈向思考最优：大语言模型推理测试时计算的扩展。载于第三十九届神经信息处理系统年会。https://openreview. net/forum?id=6ICFqmixlS[70] Shunyu Yao、Dian Yu、Jeffrey Zhao、Izhak Shafran、Tom Griffiths、Yuan Cao 和 Karthik Narasimhan。2023年。思维树：利用大语言模型进行审慎问题求解。神经信息处理系统进展 36 (2023年)，11809–11822。[71] Shunyu Yao、Jeffrey Zhao、Dian Yu、Nan Du、Izhak Shafran、Karthik R Narasimhan 和 Yuan Cao。2023年。ReAct：在语言模型中协同推理与行动。载于第十一届国际学习表征会议。https://openreview.net/forum?id=WE_vluYUL-X [72] Pengcheng Yin、Graham Neubig、Wen-tau Yih 和 Sebastian Riedel。2020年。TaBERT：对文本与表格数据进行联合理解的预训练。载于第*58*届计算语言学协会年会论文集。8413–8426。[73] Longhui Yu、Weisen Jiang、Han Shi、Jincheng YU、Zhengying Liu、Yu Zhang、James Kwok、Zhenguo Li、Adrian Weller 和 Weiyang Liu。2024年。MetaMath：为大语言模型自举生成数学问题。载于第十二届国际学习表征会议。https://openreview. net/forum?id=N8N0hgNDRt [74] Tao Yu、Zifan Li、Zilin Zhang、Rui Zhang 和 Dragomir Radev。2018年。TypeSQL：基于知识的类型感知神经文本到 SQL 生成。载于*2018*年北美计算语言学协会年会论文集：人类语言技术，第*2*卷（短文）。588–594。[75] Tao Yu、Rui Zhang、He Yang Er、Suyi Li、Eric Xue、Bo Pang、Xi Victoria Lin、Yi Chern Tan、Tianze Shi、Zihan Li、Youxuan Jiang、Michihiro Yasunaga、Sungrok Shim、Tao Chen、Alexander Fabbri、Zifan Li、Luyao Chen、Yuwen Zhang、Shreya Dixit、Vincent Zhang、Caiming Xiong、Richard Socher、Walter Lasecki 和 Dragomir Radev。2019年。CoSQL：面向跨领域数据库自然语言接口的对话式文本到 SQL 挑战。载于*2019*年自然语言处理实证方法会议与第九届国际自然语言处理联合会议论文集。计算语言学协会。[76] Tao Yu、Rui Zhang、Kai Yang、Michihiro Yasunaga、Dongxu Wang、Zifan Li、James Ma、Irene Li、Qingning Yao、Shanelle Roman 等。2018年。Spider：用于复杂跨领域语义解析和文本到 SQL 任务的大规模人工标注数据集。载于*2018*年自然语言处理实证方法会议论文集。3911–3921。[77] Tao Yu、Rui Zhang、Michihiro Yasunaga、Yi Chern Tan、Xi Victoria Lin、Suyi Li、Heyang Er、Irene Li、Bo Pang、Tao Chen、Emily Ji、Shreya Dixit、David Proctor、Sungrok Shim、Jonathan Kraft、Vincent Zhang、Caiming Xiong、Richard Socher 和 Dragomir Radev。2019年。SParC：上下文中的跨领域语义解析。载于第*57*届计算语言学协会年会论文集，Anna Korhonen、David Traum 和 Lluís Màrquez（编）。计算语言学协会，意大利佛罗伦萨，4511–4523。doi:10.18653/v1/P19- 1443[78] Eric Zelikman、Yuhuai Wu、Jesse Mu 和 Noah Goodman。2022年。STaR：通过推理自举提升推理能力。神经信息处理系统进展 35 (2022年)，15476–15488。

[79] Dan Zhang、Sining Zhoubian、Ziniu Hu、Yisong Yue、Yuxiao Dong 和 Jie Tang。2024年。Rest-mcts*：通过过程奖励引导的树搜索进行大语言模型自训练。神经信息处理系统进展 37 (2024年)，64735–64772。[80] Wenlong Zhang、Chengyang He、Guanqun Yang、Dipankar Bandyopadhyay、Tian Shi 和 Ping Wang。2024年。利用大语言模型在特定领域 NoSQL 数据库上进行自然语言查询。载于*2024*年 *IEEE* 生物信息学与生物医学国际会议（*BIBM*）。IEEE，5174–5181。[81] Wenlong Zhang、Kangping Zeng、Xinming Yang、Tian Shi 和 Ping Wang。2023年。文本转ESQ：一种两阶段可控方法，用于从 NoSQL 数据库高效检索疫苗不良事件。载于第*14*届 *ACM* 生物信息学、计算生物学与健康信息学国际会议论文集。1–10。[82] Penghao Zhao、Hailin Zhang、Qinhan Yu、Zhengren Wang、Yunteng Geng、Fangcheng Fu、Ling Yang、Wentao Zhang、Jie Jiang 和 Bin Cui。2026年。面向人工智能生成内容的检索增强生成：综述。数据科学与工程 (2026年)，1–29。[83] Yaowei Zheng、Richong Zhang、Junhao Zhang、YeYanhan YeYanhan 和 Zheyan Luo。2024年。LlamaFactory：语言模型统一高效微调。载于第*62*届计算语言学协会年会论文集（第*3*卷：系统演示）。400–410。[84] Victor Zhong、Caiming Xiong 和 Richard Socher。2017年。Seq2sql：利用强化学习从自然语言生成结构化查询。*arXiv* 预印本 *arXiv:1709.00103* (2017年)。

初步实验与方法详情 A.1 小型语言模型在 NoSQL 数据库上的推理能力

本节采用 Qwen2.5-Coder-7B-Instruct 作为骨干模型。表5显示，在中间推理步骤的辅助下，小型语言模型生成 NoSQL 查询的性能低于直接生成此类查询的小型语言模型。这一现象可归因于小型语言模型固有地缺乏 NoSQL 专属推理能力。Zero-Shot-Direct 指小型语言模型直接生成 NoSQL 查询的设置，而 Zero-Shot-CoT 则指模型仅在生成中间推理步骤后才生成 NoSQL 查询的设置。提示设计详情见图14。

表5：小型语言模型面向 NoSQL 的推理能力初步实验。

	基于查询			基于执行		
方法	EM	QSM	QFC	EX	EFM	EVM
Zero-Shot-Direct	0.0882	0.4232	0.5580	0.3905	0.4771	0.5008
Zero-Shot-CoT	0.0376	0.2868	0.5016	0.3194	0.3775	0.3913

A.2 其他方法细节

A.2.1 路径优化。 通过小语言模型进行路径优化的提示词见图8。具体而言，我们仅保留 <step> 与 <draft> 标签内的内容。<reasoning> 标签中的内容会被丢弃，因为它仅用于提升小语言模型在优化这些路径时的性能，后续处理并不需要。为提高训练效率，我们将top-k 选择设为top-1。

A.2.2 自训练流程。 基于我们大量的实验，小语言模型在每一轮都从初始基础模型进行微调，而不是在上一轮的模型上增量训练，以降低过拟合风险。该训练设计与近期研究 [14]的发现一致。与三阶段监督微调过程相对应的详细提示模板分别见图9 和图 10。

B 补充数据集详情

B.1 CoSQL 数据集

本实验使用了 CoSQL，这是一个广受欢迎且复杂的跨领域设置下的对话文本到 SQL 语料库。跨领域设置是指训练集、开发集和测试集中的特定领域（例如餐厅和航班）各不相同的情况。我们利用 CoSQL 的训练集和开发集构建数据集。值得注意的是，CoSQL 的开发集被视为 CoNoSQL 的测试集。这些集合共包含 166 个数据库和 2，459 段对话。每段对话都包含多个用户问题及其对应的 SQL 查询。对话轮次（即问题-SQL 对）的数量为 10，802。

B.2 数据集构建

表 6 清晰地展示了所提出的数据集与以往相关研究中使用的数据集之间的主要差异。尤其是，与现有的对话文本到 SQL 数据集（例如 SParC 和 CoSQL）不同，所提出的数据集聚焦于一种比 SQL 数据库预定义结构更灵活的模式。此外，对于复杂问题，以往数据集中的 SQL 查询通常涉及关系数据库中的跨表连接。相比之下，NoSQL 查询通过在 NoSQL 数据库中查询带有嵌套字段的单个文档，简化了这一过程。图12展示了用于细化和重新生成所提出数据集 CoNoSQL 的提示词。

在方法的设置方面，采用 GPT-4o、DeepSeek-R1 [16] 和 Llama3-70B [9] 作为大语言模型的代表。相似对话的数量设为 3，且每段对话都应彼此不同。对于问题的嵌入，本实验采用了 HuggingFace Sentence Transformers 库 [65]中的预训练模型 all-MiniLM-L6-v2。

B.3 质量控制

为进一步确保 CoNoSQL 数据集的高质量，我们采用后处理流程来提升数据质量。对于训练集，移除了执行结果与对应 SQL 的执行结果不匹配的查询。对于测试集，则采用人工干预 [6, 62] 。具体而言，所有查询均经过人工检查，评估指标不仅包括执行准确率，还包括语义等价性。例如，考虑一个询问“我可以查看客户列表吗？”的用户问题。SQL 查询“Select * From Customers”和转换后的 NoSQL 查询“db.Customers.find()”会被重新标注为语义等价，尽管二者的执行结果不同，这是由于 SQL 和 NoSQL 数据库中固有的去重模式不同所致。因此，有必要通过人工评估对转换后的 NoSQL 查询重新标注。评估标准如下。

- 连贯性（是或否）： NoSQL 查询是否与对话上下文相关且连贯?
- 忠实性（是或否）： NoSQL 查询的输出是否与用户问题一致?

经过人工评估后，专家（即论文作者）会纠正剩余的错误 NoSQL 查询。人工纠正总结如下。

- 首选使用聚合方法，例如“db.teacher.aggregate([{ $ count: "count" }])”，来统计选中文档的数量，因为如果选中文档数量为 0，语义等价的 NoSQL 查询“db.teacher.countDocuments()”的执行结果将是空集，而不是“0”。
- 使用条件“preserveNullAndEmptyArrays:true”是为了避免某些情况下执行结果为空数组所导致的错误。
- 在使用“sum”和“count”等聚合运算符之前，会使用“$ toDouble”运算符来避免计数错误。
- 人工审查时会考虑语义等价性，即使 SQL 查询的执行结果与转换后的 NoSQL 查询的执行结果不同。

表 6：CoNoSQL 与其他现有文本到查询数据集的比较。

数据集	数据库类型	目标语言	构建	多轮
VAERSESQ [81]	ElasticSearch	NoSQL	合成	No
TEND [36]	MongoDB	NoSQL	人工+大语言模型	No
MultiTEND [46]	MongoDB	NoSQL	人工+大语言模型	No
SParC [77]	SQLite	SQL	人工	Yes
CoSQL [75]	SQLite	SQL	人工	Yes
CoNoSQL（我们的方法）	MongoDB	NoSQL	人工+大语言模型	Yes

字段和文档的数量			
# 字段	平均字段数/数据库数	最大字段数	最小字段数
2,475	15.09	53	2
# 文档	平均文档数/数据库数	最大文档数	最小文档数
132,469	807.74	44,139	0

表7：数据库统计信息

查找方法中的操作				
投影数	匹配数	# 排序	# 限制	# 其他
2,003	1,383	370	319	108
聚合方法中的阶段				
# 展开	# 投影	# 匹配	# 分组	# 排序
9,277	7,374	5,649	4,160	1,548
# 查找	# 限制	# 计数	# 其他	
1,491	1,394	780	3,230	

表8：方法统计

B.4 数据集统计与分析

本节讨论 CoNoSQL 数据集的统计信息。表7总结了 NoSQL 数据库的统计信息。每个数据库的字段平均数为 15.09 个，每个数据库的文档平均数超过 800 个，表明这些数据库具有较高的复杂度。表8汇总了训练集和测试集中 NoSQL 查询所使用的方法的统计信息。结果显示，CoNoSQL 数据集的 NoSQL 查询所用的操作/阶段覆盖范围广且多样。

C 补充实验与细节

C.1 实现细节

C.1.1 基线方法。 免训练范式所使用的提示词详见图13。对于监督微调基线方法，直接NoSQL预测遵循 LLaMA-Factory [83] 提供的 Alpaca 格式。引导式 STaR-SFT 方法所使用的提示词详见图14。对于单轮文本到NoSQL模型 SMART，我们将用户问题与对话历史拼接为一个统一的文本序列。随后使用该序列构建提示词，并将其输入语言模型进行处理。基于智能体的方法使用 LangChain [1] 平台实现，其对应的提示词分别详见图16和图17。

[1]https://www.langchain.com/

C.1.2 超参数。 对于免训练范式，与大多数研究 [4, 25, 68]一样，采用了 0.0 的温度值。如果输入文本超出模型要求的最大令牌限制，则会从文本开头开始截断。少样本学习和检索增强生成方法采用十样本设置。在少样本学习设置中，随机选择相似示例。对于监督微调（SFT）方法，训练轮数为 3，批大小为 32。我们使用 AdamW优化器和余弦学习率调度器，初始学习率设为“1.0e-5”。Stage-MCTS 的详细配置见表9。

表9：Stage-MCTS 中使用的超参数

超参数	值
c	1.414
m	3
max_depth	8
$N_{rollout}$	10
顶部-k	1

C.2 指标

指标的详细定义如下。

C.2.1 精确匹配（*EM*）。EM 用于评估预测查询是否与真实值查询精确匹配。计算公式为

$$EM = \frac{N_{em}}{N},$$

其中， N_{em} 表示与真实值完全匹配的查询数量， N 表示测试集中查询的总数。

C.2.2 查询阶段匹配（*QSM*）。QSM 用于评估预测的 NoSQL 查询中的关键阶段是否与真实值 NoSQL 查询中的关键阶段相符，包括阶段顺序和阶段关键词。这些阶段包括匹配、分组、查找等。计算公式为

$$QSM = \frac{N_{qsm}}{N},$$

其中 N_{qsm} 定义为查询的数量，即预测查询的关键阶段与真实查询中的相应阶段完全匹配的查询数量。

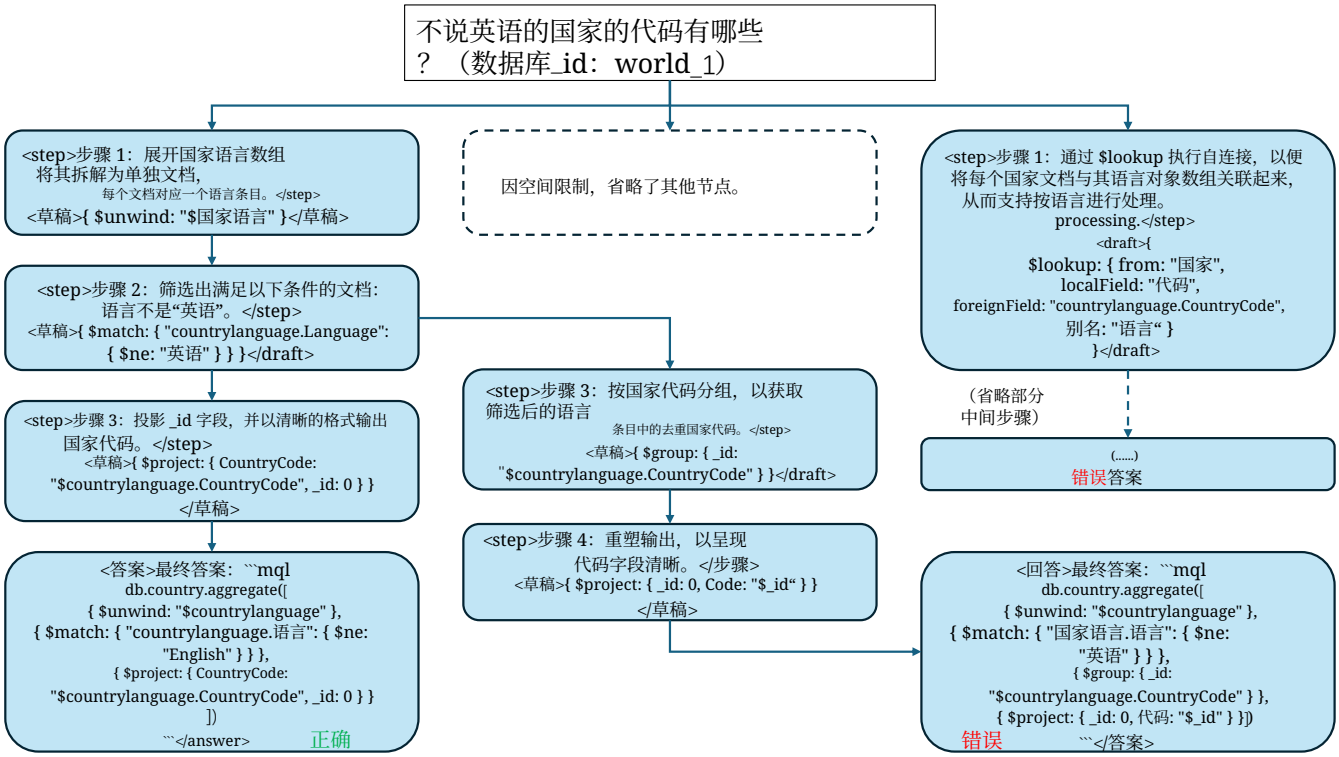


图7：基于MCTS的 NoSQL 查询生成示例。该补充图展示了我们的方法如何在给定相同先前步骤的情况下枚举多个候选阶段，然后基于奖励模型进行引导式搜索。

C.2.3 查询字段覆盖率（*QFC*）。 QFC用于评估预测查询中的字段是否全面覆盖真实查询中出现的所有字段。这包括数据库字段和查询定义的字段。计算公式为

$$QFC = \frac{N_{qfc}}{N},$$

其中， N_{qfc} 表示完全覆盖相应真实查询中所有字段的查询数量。

C.2.4 执行准确率（*EX*）。EX旨在衡量预测查询在数据库上的执行结果的准确性。计算公式为

$$EX = \frac{N_{ex}}{N},$$

其中， N_{ex} 表示执行结果与对应真实查询的执行结果完全相同的查询数量。

C.2.5 执行字段匹配（*EFM*）。EFM用于评估预测查询的执行结果中的字段名称是否与预期结果（即真实查询的执行结果）中的字段名称完全一致。计算公式为

$$EFM = \frac{N_{efm}}{N},$$

其中， N_{efm} 表示执行结果中字段名称完全匹配的查询数量。

C.2.6 执行值匹配（*EVM*）。 EVM 用于评估预测查询的执行结果中的值是否与真实查询的执行结果中的值匹配。计算公式为

$$EVM = \frac{N_{evm}}{N},$$

其中， N_{evm} 表示执行结果中值完全匹配的查询数量。

表10：自训练各轮次中各模块的消融研究。

	基于查询			基于执行		
方法	EM	QSM	QFC	EX	EFM	EVM
第1轮						
StageMCTS	0.1250	0.4935	0.6520	0.5270	0.6201	0.6675
不使用 Reward-PR	0.0596	0.3734	0.5572	0.4183	0.4943	0.5605
不含 MCTS-TTS	0.1176	0.4796	0.6438	0.4894	0.6005	0.6315
第2轮						
StageMCTS	0.1397	0.5106	0.6503	0.5433	0.6266	0.6806
不含 Reward-PR	0.0694	0.3987	0.5923	0.4502	0.5335	0.6087
不含 MCTS-TTS	0.1307	0.4886	0.6462	0.5082	0.6185	0.6511
第3轮						
StageMCTS	0.1462	0.5221	0.6552	0.5392	0.6266	0.6838
不使用 Reward-PR	0.0629	0.3930	0.5792	0.4526	0.5368	0.6176
不使用 MCTS-TTS	0.1413	0.5123	0.6479	0.5180	0.6225	0.6601

表 11：从测试集中选取的问题所对应的不同方法的查询及结果案例研究。绿色表示正确的答案与真实值匹配。红色表示错误的答案及错误结果。

方法	查询语句与结果
对话历史	省略了 2 轮
问题（ID：215）	最多接受过 3 项治疗的专业人员有多少名?
真实值	Professionals 集合.aggregate([{ $unwind: "$Treatments" }, { $group: {_id: "$professional_id", ↪ 治疗计数：{ $sum: 1 } } }, { $match: { treatmentCount: { 小于等于: 3 } } }, { $group: { _id: null, ↪ 计数: { 求和: 1 } } }, { $project: { _id: 0, 计数：1 } }]) **结果：['计数'： 8]**
Stage-MCTS（我们的方法）	Professionals 集合.aggregate([{ $unwind: "$Treatments" }, { $group: {_id: "$professional_id", 治疗计数: ↪ { $sum: 1 } } }, { $match: { 治疗计数：{ $lte: 3 } } }, { $count: "profsWithFewTreatments" }]); **结果：['profsWithFewTreatments'： 8]**
零样本（GPT-4o）	Professionals 集合.aggregate([{ $addFields: { treatmentCount: { $size: "$Treatments" } } }, { $match: { ↪治疗计数: { $lte: 3 } } }, { $count: "professionalsWithAtMost3Treatments" }]); **结果：['professionalsWithAtMost3Treatments'： 15]**
STaR-SFT（DeepSeek-R1 数据）	Professionals 集合.aggregate([{ $addFields: { treatmentCount: { $size: "$Treatments" } } }, { $match: { ↪treatment计数: { $小于等于: 3 } } }, { $计数: "专业人员_的_数量" }]) **结果：['数量_的_专业人员'： 15]**

C.3 附加消融研究

自训练流程的评估。表 10 展示了对所有自训练轮次中各个模块进行的详细消融研究。表中显示，移除任何单个组件都会导致车型在所有指标上的性能下降，这证实了每个模块都对整体性能有积极贡献。值得注意的是，随着轮次增加，所有车型的性能均有所提升，证明了所提出的自训练流水线的有效性。

测试时缩放策略的评估。本消融研究旨在评估奖励模型在测试时缩放策略中的有效性。主要结果见表 12。在 Stage-MCTS 中，自监督奖励模型（SRM）与蒙特卡洛树搜索（MCTS）相结合。作为比较，在与所提方法相同的搜索预算下，也采用 Self-Consistency 和 Best-of-N（BoN） [33]作为缩放方法，其中 Self-Consistency 使用与我们相同的 EVM 指标。本消融研究中使用的过程奖励模型（PRM）与 [79]中的模型相同。具体而言，推理路径被划分为正确和错误两类，并分别赋予 1.0 和 -1.0 的真实标签。PRM 训练被建模为分类任务，模型则通过在策略小语言模型上附加一个线性层构建，从而将输出概率直接映射为标量奖励值。值得注意的是，这些结果验证了所提方法的有效性：SRM 简单却十分有效。两种采用 PRM 增强的测试时缩放方法，其性能均低于未采用任何测试时缩放的基线，表明 PRM 无法准确判断生成的 NoSQL 查询是否正确，这是由 NoSQL 生成本身的复杂性所致。对于 Self-Consistency，它主要提升了基于查询的性能，但其基于执行的指标低于结合 SRM 的 MCTS。

表 12：基于经过三轮自训练的小语言模型，对测试时扩展策略进行的消融研究。

	基于查询			基于执行		
方法	EM	QSM	QFC	EX	EFM	EVM
无 TTS	0.1413	0.5123	0.6479	0.5180	0.6225	0.6601
自一致性	**0.1503**	**0.5319**	0.6528	0.5319	0.6111	0.6650
BoN + PRM	0.1291	0.4951	0.6283	0.4935	0.5817	0.6348
蒙特卡洛树搜索 + PRM	0.1405	0.5245	0.6487	0.5180	0.6119	0.6495
蒙特卡洛树搜索 + SRM（本文）	0.1462	0.5221	**0.6552**	**0.5392**	**0.6266**	**0.6838**

C.4 补充案例研究

图7表明，Stage-MCTS能够在先前步骤相同的情况下枚举多个候选阶段，然后基于奖励模型执行引导式搜索。

表 11 和表13给出了具体示例，展示了单路径基线方法和我们提出的基于树的搜索方法 Stage-MCTS 生成的 NoSQL 查询及相应执行结果。具体而言，表 11 展示了在生成简单查询时，Stage-MCTS 如何仅使用单个 $count 运算符来简化聚合管道。相比之下，GPT-4o 增强的零样本方法犯了与 STaR-SFT 相同的错误：它返回了 15 位专业人员，而正确值应为 8。两种方法都错误地对 Treatments 数组应用了 $size，因而无法准确反映治疗的实际数量。表13显示，在生成复杂查询时，Stage-MCTS 遵循 MongoDB 最佳实践并采用了适当的操作符。然而，GPT-4o 增强的零样本方法因数据转换错误而失败，表明它无法处理潜在的数据质量问题。至于 STaR-SFT，尽管它得出了正确答案（即 'amc'），但也额外返回了错误答案（即 'renault' 和 'ford'），并依赖不必要的复杂聚合管道，其中包含冗余的分组和投影阶段。

表13：从测试集中选取的一个问题的查询及不同方法对应结果的案例研究。绿色表示正确且与真实值一致的答案。红色表示错误存在执行错误或结果不正确的答案。橙色表示部分正确但存在一些问题的答案。

方法	查询语句和结果
对话历史	省略5轮
问题（ID：182）	对于气缸数为 4 的汽车，哪种车型的马力最大?
真实值	db.continents.聚合([{ $unwind: "$countries" }, { $unwind: "$countries.car_makers" }, { $unwind: "↪国家.car_makers.车型_list" } , { $unwind: "$国家.car_makers.车型_list.car_names" } , { ↪ $unwind: "$国家.car_makers.model_list.car_names.cars_数据" }, { $match: { $expr: { $and: [{ $eq: ↪ ["$国家.car_makers.model_list.car_names.cars_数据.Cylinders", 4] }, { $eq: ["$国家.car_makers. ↪ 车型_list.car_names.制造商 ID" , "$countries.car_makers.model_list.car_names.cars_数据.Id"] }] } } } , { ↪$排序: { "countries.car_makers.model_list.car_names.cars_data.马力": -1 } } , { $限制: 1 } , { ↪$project: { _id: 0, 车型: "$countries.car_makers.model_list.car_names.车型" } }]) **结果：['车型': 'amc']**
Stage-MCTS（我们的方法）	db.continents.聚合([{ $unwind: "$countries" }, { $unwind: "$countries.car_makers" }, { $unwind: "↪国家.car_makers.车型_list" } , { $unwind: "$国家.car_makers.车型_list.car_names" } , { ↪$unwind: "$国家.car_makers.车型_list.car_names.汽车_数据" } , { $match: { "国家.car_makers. ↪车型_list.car_names.汽车_数据.气缸数", 4 } }, { $sort: { "国家.car_makers.车型_list.car_names. ↪ 汽车_数据.马力": -1 } }, { $limit: 1 }, { $project: { _id: 0, 车型: "$countries.car_makers. ↪model_list.car_names.车型" } }]); **结果：['车型': 'amc']**
零样本 (GPT-4o)	db.continents.聚合([{ $unwind: "$countries" }, { $unwind: "$countries.car_makers" }, { $unwind: "↪国家.car_makers.车型_list" } , { $unwind: "$国家.car_makers.车型_list.car_names" } , { ↪$unwind: "$countries.car_makers.model_list.car_names.cars_数据" } , { $match: { "国家.car_makers. ↪车型_list.car_names.汽车_数据.气缸数": 4 } }, { $project: { "车型": "$国家.car_makers. ↪ 车型_list.car_names.车型" , "马力": { $toInt: "$国家.car_makers.model_list.car_names. ↪ 汽车_数据.马力" }, "_id": 0 }}, { $sort: { "马力": -1 } }, { $限制: 1 }]) **结果：查询执行错误：查询_错误：聚合期间计划执行器出错 :: 原因是 :: 在 $convert 中解析数字 'null' 失败，且未设置 onError 值：Did 未消耗整个字符串。**
STaR-SFT（DeepSeek-R1 数据）	db.continents.聚合([{ $unwind: "$countries" }, { $unwind: "$countries.car_makers" }, { $unwind: "↪国家.car_makers.车型_list" } , { $unwind: "$国家.car_makers.车型_list.car_names" } , { ↪$unwind: "$countries.car_makers.model_list.car_names.cars_数据" } , { $match: { "国家.car_makers. ↪model_list.car_names.cars_data.气缸数": 4 } }, { $project: { model: "$countries.car_makers. ↪ model_list.car_names.车型" , 马力: "$国家.汽车_制造商.车型_列表.汽车_名称.汽车_数据. ↪ 马力" } }, { $group: { _id: null, max马力: { $max: "$马力" }, 所有文档: { $push: " ↪ $$ROOT" } } } , { $unwind: "$allDocs" }, { $match: { $expr: { $eq: ["$allDocs.马力" , " ↪ $maxHorsepower"]} } }, { $project: { _id: 0, 车型: "$allDocs.model" } }, { $group: { _id: "$Model", ↪ 车型: { $first: "$车型" } } }, { $project: { _id: 0, 车型: 1 } }]) **结果：['车型': 'renault' , '车型': 'amc' , '车型': 'ford']**

D 提示模板

本研究中使用的提示模板详述如下。

```
你是一名 MongoDB 聚合专家。你的任务是分析给定的最终解答，并重构得出该解答的思考过程。


** 工作流程：**
1. **分析最终解答：**理解完整的聚合管道 2. **拆解流水线：**将流水线分解为逻辑步骤 3. **逐步推理：**针对每




** 输出格式（严格遵循）：**

<思考 >
<step>Step 1: [Anal分析w ha在此 s st阶段成果 lish以及 d why it'可能 itioned here]</step>
<草稿 >[复制此步骤对应的 MongoDB 聚合阶段]</草稿 ><推理 >[说明其目的、逻辑以及任何性能考量]</推理 >


<步骤 >步骤 2: [分析此阶段的作用，以及为何将其放在此处]</step ><草稿 >[复制此步骤中的 MongoDB 聚合阶段



...
<步骤 >步骤 N: [分析此阶段的作用，以及为何将其放在此处]</step ><草稿 >[复制此步骤中的 MongoDB 聚合阶段



** 要求：**
- 将给定的最终解答拆解为逻辑步骤 - 每个步骤都应包含：分析、草稿阶段和推理 - 说明每个阶段的策略目的（为何需




## 上下文信息

**数据库模式：**


**对话历史：**
{ 对话_历史 }
（注意："无历史"表示首次交互。）

**原始问题：** {QUESTION}


** 最终解答（目标查询）：** {FINAL_


**你的任务：**
分析给定的最终解答，并仅生成能够推导出该解答的思考过程。你的输出应：


1. 将聚合管道拆解为逻辑步骤 2. 对每个步骤进行说明：- 该步骤实现了什么 - 为




    ow th进展      ive flow f来自一个s tep to the next t
4. 请勿修改          解决方案或提出替代方案

** 输出格式要求：** - 仅输出包含步骤的 <think> 部分 - 每个步骤必须包含三个部分： <step>、
```

图8：路径优化提示词。

```
<单步生成>
你是一位 MongoDB 聚合专家。 逐步生成查询 以下 这些规则：

** 工作流程：**.
1 **如果不存在解决方案：** 从头开始 —— 提供 the 第一个逻辑 step
2. **If 解决方案 is 不完整：** 仅提供 下一个逻辑上的 step
3. **If 解决方案 已完成：** 仅提供 最终 答案 查询

** 输出格式（严格遵循以下一种格式）：**

1. **对于 中间步骤：**
<步骤>步骤 [编号]: 「简要 推理 关于此阶段的 目的」</步骤>
<草稿>[草稿 MongoDB 聚合阶段]</草稿>

示例：
<步骤>步骤 1: 筛选活跃用户 by 使用位置 $match </步骤>
<草稿>{ $match: { 状态: "启用" , 位置: "纽约市" } }</草稿>

2. **对于 最终 答案：**
<答案>最终 答案: ```mql
[完整聚合 流水线]
```</答案>
示例: `<答案>最终 答案: ```mql\ndb.department.聚合 ([\n { $unwind: "$instructors" }\n]);\n```</答案>`

最佳实践：
- 使用 `$match` 尽早进行筛选，以减少数据量
- 使用 `$lookup` only 必要时 需要连接时
- 使用以下方式处理数组： `$unwind` or `$group` 恰当处理
- 整形输出 with `$project` or `$addFields` at 适当的阶段
- 考虑 性能 每个阶段的影响 阶段

上下文

数据库模式：
{模式_上下文}

对话历史：
{ 对话_历史 }
（注意：轮次之间由 “|”分隔。“无 历史” 表示 首次 交互。）

** 当前任务**
问题：{QUESTION}

当前进度：
{ 现有_解决方案 }

** 提示：**
{HINT}

需要执行的操作：
分析 当前 州并 提供：
- 下一步 聚合步骤 推理, OR
- 最终的 最终 完整查询

重要提示： 仅按上述指定格式输出。
```

**图 9：下一步生成提示词。**
```

使用推理路径优化的蒙特卡洛树搜索：面向对话式文本到 NoSQL 的小型语言模型 Conference'17，2017年7月，美国华盛顿特区

```
<展开 >
你是一位 MongoDB 聚合专家。请根据现有解答，一次性生成完整查询，并说明剩余步骤。


** 工作流程：**
1. **分析问题：** 根据模式、问题和现有解决方案理解需求 2. **规划所有步骤：** 设计完整的聚合管道，并对每个阶段进行逻辑推理 3. **提



** 输出格式（请严格遵循）：**

1. **首先，输出剩余步骤及推理：**<步骤 >第 i 步 +1: [简要说明此阶段的目的]</步骤 ><


<步骤 >第 i+2 步: [简要说明此阶段的目的]</步骤 ><草稿 >[起草 MongoDB 聚合阶段]</草稿 >


<步骤 >第 i+N 步: [简要说明此阶段的目的]</步骤 ><草稿 >[起草 MongoDB 聚合阶段]</草稿 >


2. **然后，输出完整的最终答案：**
<答案 >最终答案：```MQL
[完整聚合流水线]
```</答案 >

示例输出格式：（针对现有解决方案）{EXAMPLE}

最佳实践：
- 使用 `$match` 尽早进行筛选，以减少数据
- 在必要时使用 `$lookup` only 需要连接
- 使用以下方式处理数组 `$unwind` or `$group` 恰当处理
- 整理输出 with `$project` or `$addFields` at 适当的阶段
- 考虑性能每个阶段的影响阶段
- 保持流水线逻辑清晰且高效

上下文

数据库模式：

对话历史：
{ 对话_历史 }
（注意：轮次之间以"|"分隔。"无历史"表示首次交互。）

** 当前任务**
问题：{QUESTION}

当前进度：
{ EXISTING_SOLUTION }

** 提示：**
{HINT}

需要执行的操作：
分析当前州，并提供：。
1 所有中间聚合步骤及其推理每个阶段
2. The 最终完整聚合流水线

** 重要：**
- 在 <step> 部分中输出所有步骤，并附上 <草稿 >- 在 <answer> 部分中输出完整的最终查询 - 如果现
```

**图10：展开提示词。**
```

```
<步骤 >步骤 i+1: 筛选活跃用户 by 使用位置 $match </步骤>
<草稿 >{ $match: { 状态: "active" , 位置: "纽约市" } }</草稿>

<步骤 >步骤 i+2: 与部门 集合，使用 $lookup </步骤>
<草稿 >{ $lookup: { 来自: "部门". 本地字段: "dept_id". 外部字段: "_id". as: "部门" } }</草稿>

<步骤 >步骤 i+3: 展平 部门数组 使用 $unwind </步骤>
<草稿 >{ $unwind: "$部门" }</草稿 >

<步骤 >步骤 i+4: 用户分组 by 部门和 计算平均工资 </步骤>
<草稿 >{ $group: { _id: "$department.name", 平均薪资: { $平均值: "$薪资" } } }</草稿>

<步骤 >步骤 i+5: Sort 结果 按平均工资 降序 </步骤>
<草稿 >{ $sort: { avgSalary: -1 } }</草稿>

<答案 >最终 答案: ```mql
db.users.aggregate ([
  { $match: { status: "active" , 位置: "纽约市" } },
  { $lookup: { 来自: "部门". 本地字段: "dept_id". 外部字段: "_id". as: "部门" } },
  { $unwind: "$部门" },
  { $group: { _id: "$部门.name", 平均薪资: { $avg: "$薪资" } } },
  { $sort: { 平均薪资: -1 } }
]);
```</答案 >
```

**图 11：展开提示中使用的一个示例。**

```
你是一个翻译系统 旨在将 SQL 转换为 查询 into MongoDB 查询。 你将 获得 详细信息
关于一个 MongoDB 集合, 用户的自然语言 描述其 data 检索 需求, 以及 SQL 查询 that
表示此需求。 你的任务是分析 SQL 查询并翻译 将其转换为等效的 MongoDB 查询。

以下是 the 详情 你将 内容:

MongoDB 集合及其 字段: { COLLECTION_INFO }

用户自然语言问题: {QUESTION}

对应的 SQL 查询: {SQL_QUERY}

让我们思考 step by 步骤。 使用 the 信息 提供的, 仔细 转换 SQL 查询 转换为 MongoDB 查询 that
检索 相同的 数据。 考虑所有 的各个方面 SQL 查询, such 作为筛选条件, 排序, and 连接
条件, and 反映这些 in 你的 MongoDB 查询 翻译。

提供 你的回答应按 以下格式 给出:

解释: （关于 此 MongoDB 查询）
答案:
###
（MongoDB 查询）
###

{ 对话_示例 }
```

图12：用于数据集重新生成的提示词。
```

```
你是一个 乐于助人的 助手，可以回答问题并协助完成任务， 尤其 对于相关任务 与.....相关的
MongoDB 数据库。

你将获得 一条自然语言 查询， a 数据库模式， 以及一份 .....的列表 先前 交互。
你需要回答自然语言查询 基于 于 数据库模式和 上一页 交互。
具体而言， the 历史 交互是 the 对话 历史 之间 用户和助手。 每一轮次 is
由 一个特殊的 标记 "|". 如果 没有 历史， it 将会 "No 历史 提供。 This is the 第一轮。”

自然语言查询：{QUESTION}
数据库模式：{ SCHEMA_CONTEXT}
之前 交互：{DIALOGUE_HISTORY }

你应该 only 输出 MQL 查询， 没有其他文本。 你最好确定答案是正确的。
The MQL 格式 查询 is:
```mongodb
db.collection.aggregate ();
```
or
```mongodb
db.collection.find ();
```

{EXAMPLES}
```

图13：免训练范式的提示词。

```
你是一位经验丰富的数据库专家。
现在，你需要生成一个 MongoDB 查询，给定 数据库 信息，自然语言查询和 一些额外的信息。
The 数据库结构 is 由以下内容定义 the 以下 数据库模式（一个 字典 of 集合，他们的 字段及其类型）。
The 对话 历史 是 对话 历史 用户和助手之间。 每个轮次 由一个特殊标记 "|". 如果没有 没有 历史， it 将会 "No 历史 提供。 This is the 第一轮。"

给定 数据库模式，the `自然语言查询` 以及 the `对话 历史 `。 您将 给定 数据库模式和你需要 了解 数据库和字段。

你将使用一种方法 称为 "递归 MongoDB 查询的拆分-征服方法 查询生成 从自然语言 ".

Here is a 步骤的高级描述。
1. **拆分（使用伪 MongoDB 分解子问题）：**复杂的自然语言 查询 会被递归地 拆解 down 为更简单的子问题。 Each 子问题 针对特定的 最终 MongoDB 查询所需的一项信息或逻辑 最终 MongoDB 查询。
2. **征服（子问题的真实 MongoDB）：** 对于每个 子问题（以及最初的主 问题 ）， a “伪 MongoDB” 片段被 构造。 此伪 MongoDB 表示 预期的 MongoDB 逻辑，但可能包含 占位符 针对拆分后子问题的答案。
3. **合并（重新组装）：**一旦所有子问题都得到解决 及其相应的 MongoDB 片段被 生成后 , the 该过程反向进行。 The MongoDB 片段会递归地 组合 替换 the 占位符 伪 - MongoDB 中的实际 由较低层级生成的 MongoDB。
4. **最终输出：** 这 自底向上的组装 最终形成 完整且 正确的 MongoDB 查询 能够回答原始 复杂的 问题。

数据库管理员 说明 （违反 any 以下 内容 will 将导致 可受处罚的 to 死亡！）:
{ 管理员_指令 }

重复 the 问题和提示， and 生成 the MongoDB 查询 使用递归 分而治之。

请按以下方式回复 with XML 标签结构化 as 如下:
```xml
<思考 >
你的详细 推理 用于 MongoDB 查询生成， 采用递归 拆分与征服方法。
</思考 >
<MQL >
The 最终 MongoDB 查询 能够回答问题的 The MongoDB 的格式 查询 is:
db.collection.aggregate ();
or
db.collection.find ();
</mql >
```

数据库模式：
{模式 上下文}

对话历史：
{ 对话_历史 }
（注意：轮次之间用 “|”。“无 历史” 表示 首次 交互。）

** 当前任务**
问题：{QUESTION}

** 提示：**
{HINT}

Only 输出 XML 格式 （开始 with ```xml and 结束 with ```）别名 你的回答。
```

图14：STaR-SFT的提示词。

1. **投影：** - 仅使用 `$project` 投影用户问题中提到的字段。 - 避免不必要的字段或值。 2. **聚合操作：** - 在使用 `$max` 或 `$min` 之前，

图15：STaR-SFT 中使用的数据库管理员指令。

你是 MongoDB 数据库中的一名乐于助人的助手。现在给你一个数据库模式、对话历史和自然语言查询。数据库模式是一个字典，其中包含集合名称、字段及其类

图16：ReAct智能体的提示词。

```
针对给定目标，制定一个简单的分步计划，并给出相应的 JSON 格式 MongoDB 查询子句。该计划应包含具体任务，例如“使用 MongoDB 聚合框架展开嵌套数组”。
```

图17：Plan-and-Solve 智能体的提示词。