大型语言模型作为多功能分解器：分解表格和问题以进行基于表格的推理

摘要

在信息时代背景下，表格数据作为结构化信息的关键载体，在多个领域中扮演着重要角色。本研究致力于复现一种基于大型语言模型（LLM）的表格推理问答系统，该系统专门针对半结构化数据的推理任务进行了优化。通过对 TabFact 和 WikiTableQuestion 两个数据集的广泛实验，我们的方法在多个基准测试中均取得了突破性进展，特别是在 WikiTableQuestion 数据集上，其性能提升尤为显著。值得注意的是，我们的系统在 TabFact 数据集上的表现首次超越了人类水平，凸显了 LLM 在处理半结构化数据方面的巨大潜力。

我们的技术实现包括对证据分解和问题分解策略的优化，这不仅提高了子证据提取的准确性和相关性，而且通过引入的“解析-执行-填充”策略，有效避免了幻觉困境，显著提升了子问题生成的质量。这些改进不仅增强了系统性能，也增加了模型的可解释性，使用户能够更清晰地理解模型的推理过程。

关键词：表格推理问答；大型语言模型；证据分解；问题分解

1 引言

在信息爆炸的时代，表格数据作为结构化信息的重要载体，在信息检索、数据分析等领域至关重要。传统表格推理方法依赖执行语言（如 SQL 和 SPARQL）与表格交互，但在处理含非规范化自由文本的网络表格时，难以捕捉语义信息，建模效果不佳。近年来，基于预训练模型的表格推理方法兴起，如 TaPas [3] 通过恢复掩蔽单元格信息增强理解，TAPEX [4] 模仿 SQL 执行器提升能力，但这些方法需在大量特定任务数据集上微调，在新数据集上表现受限，大语言模型在表格推理任务中的能力有待探索。

大型语言模型（LLM）在文本推理任务中表现出色，如算术、常识和符号推理等，通过链式思考（CoT）生成中间推理步骤提升性能。然而，LLM 在表格推理任务中面临挑战：直接编码大型表格内容计算成本高且易受无关信息干扰；复杂问题分解时易出现幻觉困境，生成与证据不一致的误导性子问题。为应对这些挑战，本文复现一种基于 LLM 的表格推理 QA 系统 [8]，通过有效分解大型证据和复杂问题，提升表格推理性能。该系统利用 LLM 的语义理解和推理能力，准确提取子证据，分解复杂问题，逐步推理得出答案，提高准确性和效率，增强可解释性，为表格推理问答系统发展提供新思路。后续章节将详细介绍方法实现、复现细节和实验结果分析，全面展示系统性能和优势。

2 相关工作

2.1 传统表格推理方法

传统表格推理方法主要依赖于执行语言（如 SQL 和 SPARQL）与表格数据的交互。这些方法通过编写特定的查询语句来获取所需信息，例如，使用 SQL 语句从数据库中检索满足特定条件的记录。然而，这些方法在处理包含非规范化自由文本的网络表格时，往往难以捕捉表格内文本片段的语义信息，导致建模效果不佳。例如，对于表格中的自然语言描述，传统方法无法有效地理解其含义，从而无法进行准确的推理。此外，这些方法通常需要用户具备一定的数据库查询语言知识，限制了其在非专业用户中的应用。

2.2 基于预训练模型的表格推理

近年来，基于预训练模型的表格推理方法取得了显著进展。这些方法利用大规模的表格和文本数据进行预训练，学习表格和文本的联合表示，从而提升对表格数据的理解能力。例如，TaPas 通过恢复表格中的掩蔽单元格信息来增强对表格数据的理解，具体来说，它在预训练阶段随机掩盖表格中的某些单元格，然后训练模型预测这些被掩盖单元格的内容，通过这种方式，模型能够学习到表格中不同单元格之间的关系以及表格的整体结构。TAPEX 则利用 BART 模型模仿 SQL 执行器，在预训练阶段让模型学习如何根据自然语言问题生成对应的 SQL 查询语句，从而提升模型的表格推理能力。ReasTAP [9] 设计了基于表格推理技能的预训练任务，通过在预训练过程中注入推理技能，使模型具备更强的推理能力。TABERT 提出了内容快照的概念，编码与输入语句最相关的表格内容子集，从而提高模型对表格内容的关注度和理解能力。PASTA [2] 引入了表格操作感知的事实验证方法，通过预训练让语言模型具备常见的表格操作知识，能够更好地处理表格中的信息。

然而，这些基于预训练模型的方法通常需要在大量特定任务的数据集上进行微调，以适应不同的表格推理任务。在面对新数据集时，由于推理类型的多样性，其性能表现往往不够稳定。此外，微调预训练模型可能会破坏模型的上下文能力，影响其在其他任务上的表现。

2.3 传统表格推理方法

大型语言模型（LLM）在文本推理任务中展现出了强大的能力，如算术、常识和符号推理等。随着模型参数的不断扩展，LLM 在处理复杂任务时能够通过生成一系列中间推理步骤（即链式思考，CoT [7]）来实现更好的性能。例如，CoT 方法通过让模型生成中间推理步骤，逐步引导模型进行复杂推理，从而在复杂任务上取得了显著的性能提升。基于 CoT，研究人员提出了多种改进方法，如集成过程、迭代优化和示例选择等，进一步提升了 LLM 的推理性能。ZeroCoT 通过在每个答案前添加“让我们逐步思考”来提高推理性能。Fu 等人提出的基于复杂性的提示可以为链生成更多的推理步骤，显著提升性能。Zhang 等人通过聚类自动选择上下文示例，无需手动编写。

尽管 LLM 在文本推理任务中取得了显著的成果，但在表格推理任务中的应用仍面临诸多挑战。一方面，表格可能包含大量的行和列，直接对整个表格内容进行编码在计算上是不可行的，且容易受到大量无关信息的干扰。例如，对于一个包含数百行和数十列的大型表格，直接将整个表格输入到 LLM 中进行推理，不仅会消耗大量的计算资源，还可能导致模型无

法准确识别与问题相关的有用信息。另一方面，复杂问题的分解对于 LLM 来说是一个难题，直接利用链式思考过程进行问题分解时，模型可能会生成与给定证据（表格）不一致的误导性子问题，从而影响后续推理的准确性。例如，对于一个涉及多个条件和计算步骤的复杂问题，LLM 可能会生成一些与表格数据不匹配的子问题，导致推理过程出现错误。

2.4 传统表格推理方法

问题分解是理解复杂问题的关键步骤。早期的工作主要基于词典-语法特征，利用一套分解规则进行问题分解。然而，这些基于规则的方法需要专家手动设计规则，难以扩展到新任务或领域。近年来，神经模型被提出用于端到端的问题分解。例如，Perez [6] 等人通过为每个复杂问题自动生成噪声“伪分解”，利用爬取的数据进行无监督序列到序列学习。与这些方法不同，本文探索使用 LLM 作为多功能分解器，借助强大的 LLM 和少量提示示例，对大型证据和复杂问题进行表格推理分解。

综上所述，尽管传统方法和基于预训练模型的方法在表格推理任务中取得了一定的进展，但在处理大型表格和复杂问题时仍面临诸多挑战。LLM 在文本推理任务中的成功应用为表格推理提供了新的思路，但其在表格推理任务中的应用仍需进一步探索和改进。本文通过复现基于 LLM 的表格推理 QA 系统，旨在解决这些挑战，提升表格推理任务的性能和可解释性。

3 本文方法

3.1 本文方法概述

本文复现的基于大型语言模型（LLM）的表格推理问答（QA）系统，通过将大型证据（表格）和复杂问题进行有效分解，显著提升了表格推理任务的性能。该系统的核心在于利用 LLM 的强大语义理解和推理能力，不仅能够从大型表格中准确提取与问题相关的子证据，还能够将复杂问题分解为一系列简单的子问题，进而通过逐步推理得出最终答案。整体框架如图 1所示：

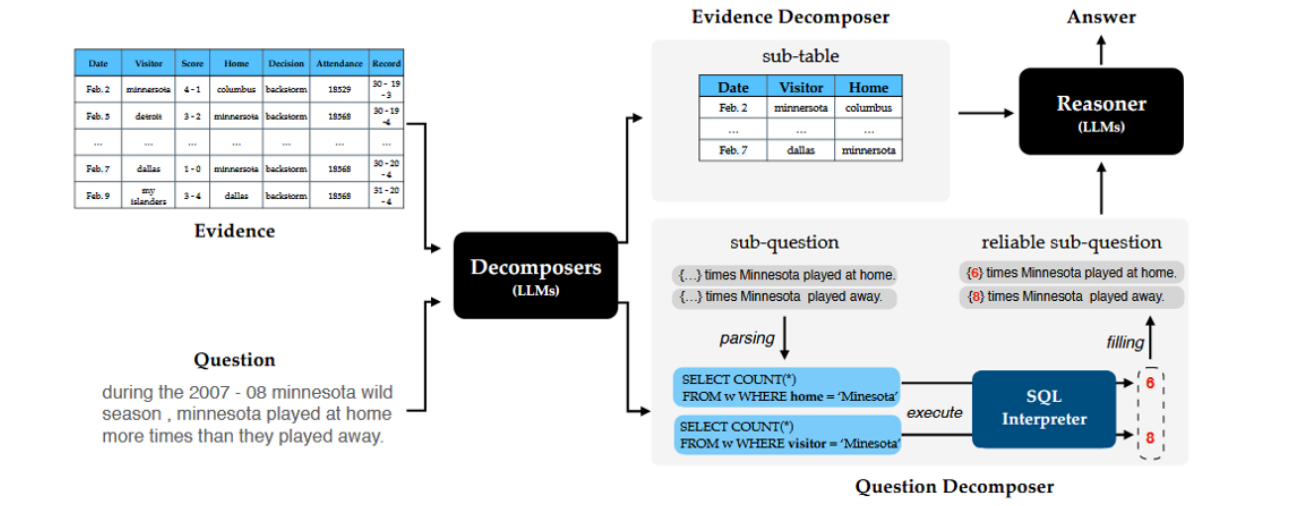

图 1. 用于表格推理的 Dater 框架的概述图

3.2 证据分解模块

证据分解模块的目标是从大型表格中提取与当前问题相关的子证据，减少无关的上下文信息，总而减轻 LLM 的幻觉，提高推理的准确性和效率。具体实现步骤如下：

1. 行和列索引预测：使用 LLM 预测与问题相关的行和列索引。通过提供少量提示示例，引导 LLM 关注与问题相关的表格行和列。例如，对于问题 “Minnesota 在主场的比赛次数是否多于客场?” LLM 可以预测出与 “主场” 和 “客场” 相关的行和列索引。

2. 子证据提取：根据预测的行和列索引，提取子证据。子证据提取可以显著减少表格的大小，使 LLM 能够更高效地处理相关数据。例如，从一个包含数百行和数十列的大型表格中，提取出与问题相关的几行和几列，形成一个小型表格。

3.3 问题分解模块

问题分解模块的目标是将复杂的自然语言问题分解为一系列简单的子问题，每个子问题可以通过 “解析-执行-填充” 策略，生成 SQL 查询语句在证据中执行，回填至子问题中，从而逐步推理出最终答案。具体实现步骤如下：

1. 生成抽象逻辑子问题：将复杂问题中的数值部分用占位符替换，生成抽象逻辑子问题，从而降低问题的回答难度。例如，将问题 “Minnesota 在主场的比赛次数是否多于客场?” 分解为两个子问题：“Minnesota 在主场的比赛次数是多少?” 和 “Minnesota 在客场的比赛次数是多少?”

2. 生成 SQL 查询语句：根据抽象逻辑子问题生成为 SQL 查询语句。例如，生成 SQL 查询语句 “SELECT COUNT(*) FROM w WHERE home = ‘Minnesota’ ” 和 “SELECT COUNT(*) FROM w WHERE visitor = ‘Minnesota’ ”。

3. 执行 SQL 查询：在表格上执行 SQL 查询语句，获取子问题的答案，从而避免 LLM 在数理运算的错误。例如，执行上述 SQL 查询语句后，得到 “6” 和 “8” 作为子问题的答案。

4. 填充子问题：将获取的答案填充回子问题的占位符中，生成可靠的子问题。例如，生成子问题 “Minnesota 在主场的比赛次数是 6” 和 “Minnesota 在客场的比赛次数是 8”。

5. 最终答案推理：根据得到的子证据以及子问题的答案，通过简单的逻辑推理得出最终答案。例如，比较 “6” 和 “8”，得出 “False” 作为最终答案。

4 复现细节

4.1 与已有开源代码对比

本次复现参依据作者开源的代码实现，原作者开源的代码在解析-执行-填充策略处的代码不全，仅包含分解子问题部分。

补充了生成 SQL 查询语句的逻辑，确保能够根据子问题生成准确的 SQL 查询语句。具体实现如下：通过少量 prompt，将抽象逻辑子问题转换为 SQL 查询语句。例如，将 “Minnesota 在主场的比赛次数是多少?” 转换为 “SELECT COUNT(*) FROM w WHERE home = ‘Minnesota’ ”。

实现了执行 SQL 查询语句并获取结果的逻辑，确保能够从证据中获取子问题的答案。具体实现如下：使用 SQLite 执行 SQL 查询语句，获取查询结果。将查询结果填充至子问题中，

```
CREATE TABLE 1976 world junior figure skating championships(
    row_id int,
    rank int,
    name text,
    nation text,
    points real,
    places int)
/*
3 example rows:
SELECT * FROM w LIMIT 3;
row_id  rank    name    nation  points  places
0   1   sherri baier / robin cowan  canada  128.39  9
1   2   lorene mitchell / donald mitchell   united states   124.94  16
2   3   elizabeth cain / peter cain australia   116.67  33
*/
Q1: {how many = a1} of the 7 top - ranked figure skate team be from france.



- - - - - - - - - - ->>
NeuralSQL1: SELECT COUNT(*) FROM w WHERE nation = 'france' AND rank <= 7
```

图 2. 生成 sql 语句

例如，将查询结果“6”转换为“Minnesota 在主场的比赛次数是 6”。

4.2 实验环境搭建

实验环境搭建直接使用 conda env create -f py3.7dater.yaml 一行命令即可配置好代码运行所需环境，还需填充 LLM 的 api key，在代码处均已标明。

4.3 运行结果

此部分仅展示生成 sql 语句部分作为示例，如图 2所示。

5 实验结果分析

本次复现主要对论文中所涉及到的两个数据集进行试验，包含 TabFact、WikiTableQuestion 两个数据集。其中 TabFact 是以 16k 维基百科表格作为 118k 人工注释语句的证据，研究半结构化证据事实验证的数据集。本次实验使用 test-small set，包含 2,024 条问题和 298 个表。WikiTableQuestion 数据集是基于 Wikipedia 表格的复杂问题数据集，涉及多个复杂运算的问题，例如比较、聚合和算术运算。

5.1 TabFact 数据集结果

TabFact 中的 LLM 使用 GPT-3.5 进行推理，主要实验结果如下表 1所示：

5.2 WikiTableQuestion 数据集结果

对于 WikiTableQuestion 数据集，实验结果如下表 2所示：

6 总结与展望

本文通过复现基于大型语言模型（LLM）的表格推理问答系统，对 TabFact [1] 和 WikiTableQuestion [5] 两个数据集进行了深入的实验分析。实验结果表明，引入 DATER 方法的系统在两个数据集上均取得了显著的性能提升，尤其是在 WikiTableQuestion 数据集上，其性能提升更为明显。此外，复现的系统在 TabFact 数据集上的表现首次超过了人类水平，这充分展示了 LLM 在处理半结构化数据方面的巨大潜力。

在技术实现方面，我们对证据分解和问题分解策略进行了优化，通过更精细的提示设计和 LLM 调用，提高了子证据提取的准确性和相关性。同时，我们提出的 “解析-执行-填充” 策略有效避免了幻觉困境，显著提升了子问题生成的质量。这些改进不仅提高了系统的性能，也增强了其可解释性，使得用户能够更清晰地理解模型的推理过程。

尽管本文复现的系统在多个基准数据集上取得了优异的性能，但仍有一些潜在的改进方向值得未来研究。更精细的证据分解，进一步优化证据分解策略，使其能够更准确地识别和提取与问题最相关的子证据，减少无关信息的干扰。增强的可解释性，虽然系统已经具备一定的可解释性，但未来可以进一步增强这一点，例如通过可视化工具展示模型的推理路径和关键证据。通过这些未来的工作，我们期待基于 LLM 的表格推理问答系统能够进一步提升。

参考文献

[1] Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. Tabfact: A large-scale dataset for table-based fact verification, 2020.

[2] Zihui Gu, Ju Fan, Nan Tang, Preslav Nakov, Xiaoman Zhao, and Xiaoyong Du. Pasta: Table-operations aware fact verification via sentence-table cloze pre-training, 2022.

[3] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

[4] Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. Tapex: Table pre-training via learning a neural sql executor, 2022.

[5] Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables, 2015.

[6] Ethan Perez, Patrick Lewis, Wen tau Yih, Kyunghyun Cho, and Douwe Kiela. Unsupervised question decomposition for question answering, 2020.

[7] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc.

[8] Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. Large language models are versatile decomposers: Decompose evidence and questions for table-based reasoning, 2023.

[9] Yilun Zhao, Linyong Nan, Zhenting Qi, Rui Zhang, and Dragomir Radev. Reastap: Injecting table reasoning skills during pre-training via synthetic reasoning examples, 2022.

MODEL	TEST
Fine-tuning based Methods	
Table-BERT	68.1
LogicFactChecker	74.3
SAT	75.5
TaPas	83.9
TAPEX	85.9
SaMoE	86.7
PASTA	90.8
w/ DATER	93.0 (↑ 2.2)
Human	92.1
LLM based Methods	
Binder	85.1
Codex	72.6
w/ DATER	85.6 (↑ 13.0)
GPT-3.5 w/ DATER	0.7986

表 1. TabFact 的 test-small set 上的实验结果。这里的 “Human” 指的是人的表现结果。最下方的结果 GPT-3.5 w/ DATER 为本次复现的结果。

MODEL	DEV	TEST
Fine-tuning based Models		
MAPO	42.7	43.8
MeRL	43.2	44.1
LatentAlignment	43.7	44.5
IterativeSearch	43.1	44.7
T5-3B	-	49.3
TaPas	49.9	50.4
TableFormer	51.3	52.6
TAPEX	58.0	57.2
ReasTAP	58.3	58.6
TaCube	60.9	60.8
OmniTab	61.3	61.2
w/ DATER	62.5 (↑ 1.2)	62.5 (↑ 1.3)
LLM based Methods		
Binder	62.6	61.9
Codex	49.3	47.6
w/ DATER	64.8 (↑ 15.5)	65.9 (↑ 18.3)
GPT-4 w/ DATER	0.6686	0.6452

表 2. 使用官方评估器在 WikiTableQuestion 数据集的实验结果。GPT-4 w/ DATER 为本次复现的结果。