大模型驱动的移动对象自然语言查询转换方法基准测试研究

王撷阳[1)] 许建秋[1)] 刘孟怡[1)] 宗辰辰[1)] 高云君[2)]

[1)](南京航空航天大学计算机科学与技术学院 南京 211106)

[2)](浙江大学计算机科学与技术学院 杭州 310027)

摘 要 近年来,移动对象数据呈现爆炸式增长,自然语言逐渐成为普通用户访问移动对象数据库的重要方式。尽管已有针对移动对象数据库自然语言接口查询系统的研究,但目前仍缺乏统一的查询转换质量评估标准。现有的跨域基准测试主要为简单的值查询,未涉及时空多维复杂查询,且未充分验证其在实际应用场景中的转换精度和效率。为此,本文提出了面向移动对象自然语言查询转换方法的基准测试并给出了基线方法。该基准测试利用大模型与人工生成相结合完成构建,并体现了查询转换的难度和复杂性:(1)覆盖科研常见的5类查询及2种真实场景查询,保证了查询类别的广泛性以及真实复杂场景下的挑战性;(2)针对自然语言查询中固有的歧义性及模糊性,显著提升了对实体抽取的挑战;(3)面对可执行语句多样且缺乏固定架构的特点,在实体、操作符与索引的组合优化以及复杂推理问题上进行了全面设计。在此基础上,本文给出了两阶段的查询转换基线方法,分别为自然语言理解和可执行语句构造。自然语言理解阶段通过从粗到细粒度的算法提取与数据库对齐的实体并进行查询类型识别。可执行语句构造阶段结合实体和操作符生成可执行的数据库查询,并利用索引优化查询效率。实验结果表明,现有方法以及大模型难以直接应用于移动对象数据库的自然语言查询转换设计,无法有效处理复杂的移动对象查询,也缺乏对查询效率的精确控制。因此,移动对象数据库的自然语言查询转换仍有较大的研究空间。基准测试采用的600个测试用例对应两类数据集:(1)真实数据集。记录了2024年10月22日南京252个移动对象全天运行的714938个轨迹点以及9900个空间信息;(2)合成数据集。基于2003年11月20日柏林562辆火车的51544个轨迹点以及8078个城市空间数据。本文提出的方法指导大模型在真实和合成数据集上的平均可翻译性及翻译精度分别为81.21%和66.14%,翻译效率分数为68.42(相对分数)。在查询转换精度和可执行语句的效率控制方面均表现优异,具有一定的参考价值。

关键词 移动对象数据库;数据库自然语言查询接口;自然语言理解;可执行语句;查询处理;大语言模型

中图法分类号 TP18 **DOI号** 10.11897/SP.J.1016.2026.01424

Large Language Model-Driven Natural Language Query Transformation Methods and Benchmark for Moving Objects Databases

WANG Xie-Yang[1)] XU Jian-Qiu[1)] LIU Meng-Yi[1)] ZONG Chen-Chen[1)] GAO Yun-Jun[2)]

[1)](School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106)

[2)](School of Computer Science and Technology, Zhejiang University, Hangzhou 310027)

Abstract Recent years have seen an explosion in moving objects data volumes with natural language becoming the preferred choice for data access. However, existing benchmarks target simple value queries over relational data, lacking unified quality metrics, support for complex spatio-temporal multi-dimensional queries, and real-world validation of translation accuracy and

收稿日期:2025-04-30;在线发布日期:2026-02-25。本课题得到国家自然科学基金联合基金项目(U23A20296)、国家自然科学基金面上项目(62472217)、江苏省研究生科研与实践创新计划项目(KYCX24_0608)资助。**王撷阳**,博士研究生,主要研究领域为移动对象数据库。E-mail: xieyang@nuaa.edu.cn。**许建秋**(通信作者),博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为时空数据管理。E-mail: jianqiu@nuaa.edu.cn。**刘孟怡**,博士研究生,主要研究领域为空间数据库。**宗辰辰**,博士研究生,主要研究领域为机器学习。**高云君**,博士,教授,国家杰出青年科学基金入选者,中国计算机学会(CCF)高级会员,主要研究领域为大数据管理与分析、DB与AI融合。

efficiency. To address this, we propose a benchmark for natural language query (NLQ) translation in moving objects databases (MODs), covering five commonly encountered moving objects queries and two real-world scenario queries. We use large language models (LLMs) and manual efforts to design the benchmark, which highlights the complexity of query transformation by: (1) encompassing diverse query types to present challenges typical of complex scenarios; (2) addressing schema linking challenges caused by natural language ambiguities; (3) offering a design framework for optimizing combinations of operators, entities, and indexes, while tackling complex reasoning problems. Our two-phase query transformation method includes natural language understanding and executable language generation. In the natural language understanding phase, a coarse-to-fine algorithm extracts entities and classifies query types to comprehend NLQ intent. In the executable language generation phase, executable queries are formed by combining entities and operators, with index optimization. Experimental results show existing methods struggle with complex MOD queries and lack precise efficiency control, underscoring significant research potential for MODs. The benchmark consists of 600 test cases across two datasets: a real dataset recording 714938 trajectory points and 9,900 spatial data from 252 moving objects in Nanjing on October 22, 2024, and a synthetic dataset comprising 51544 points and 8,078 spatial data from 562 trains in Berlin on November 20, 2003. Our method achieves average translatability, translation precision, and translation efficiency scores of 81.21%, 66.14%, and 68.42, respectively. The strong gains in precision and efficiency control offer valuable guidance for future research.

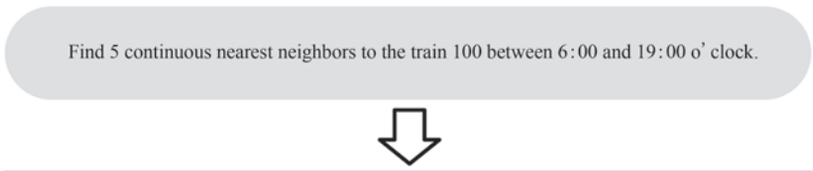
Keywords　moving objects database; natural language interface for databases; natural language understanding; executable language; query processing; large language model

1　引　言

移动对象数据的爆炸式增长使得越来越多的用户参与到移动对象数据的使用与管理中，推动了数据库自然语言查询接口（Natural Language Interface for Database，NLIDB）研究的兴起。这类接口使得非专业用户能够直接操作和管理数据，从而激活数据要素潜能，推动知识扩散并开辟经济增长新空间。同时，NLIDB的发展为移动对象数据库的自然语言查询转换提供了新思路。

移动对象数据库的文本到可执行语句的重要性。当前NLIDB研究主要面向关系型数据库，即从文本到SQL（Text-to-SQL）的转换。典型方法包括基于规则的方法如NaLIR[1]、基于神经网络的方法如ValueNet[2]以及基于大语言模型（Large Language Model，LLM）的方法如FinSQL[3]。这些方法虽取得显著进展，但缺乏对时空操作符的支持，难以有效处理移动对象的多维查询，且依赖数据库优化器解析SQL得到最优物理可执行计划，侧重翻译准确率而忽略执行效率。对于关系型数据库，现有的复杂查询如join查询等在Spider 2.0[4]中的转换成功率不足6%，更难以直接进行自然语言到物理可执行计划的转换。相比之下，时空可执行语句支持广泛的查询及操作符，能够灵活处理多种时空操作，选择合适索引直接进行查询优化。如图1所示，GPT-4o生成的SQL需要综合运用多种操作符和谓词来表示最近邻的时空语义，并利用优化器间接优化。可执行语句仅需使用*nearest_neighbor*操作符，并能选择性使用3D R-tree等索引。因此，文本到可执行语句（Text-to-executable-language，Text-to-EXE）转换研究值得进一步探索。本文的方法和实验均基于开源的可扩展时空数据库管理系统SECONDO[5]展开，后续示例均采用其支持的可执行语句。

大语言模型范式应用于文本到可执行语句的可能性。基于LLM的方法已广泛应用于Text-to-SQL的转换任务中，但缺乏专门为可执行语句设计的大规模数据集。领域知识的缺失以及幻觉问题的存在也会显著降低实体链接、操作符及索引选择的准确性。图1示例展示了由LLM生成的SQL中的实体（加粗显示），与标准实体链接结果相比存在不



	标准可执行语句	基于GPT-4o生成
转换结果	let t100 = **Trains** feed filter[.Id = **100**] extract[Trip]; query UTORdered_rtree UTOrdered feed filter [(deftime(.UTrip)intersects [const periods value (**"2020-11-20-06:00"** **"2020-11-20-19:00"**TRUE TRUE)])]nearest neighbor[UTrip, t100, **5**] consume;	SELECT A.Id FROM **Trains** AS A, **Trains** AS B WHERE B.Id=**100** AND StartPoint (Intersection(A.Trip, B.Trip)BETWEEN mpoint('**2020-11-20 06:00:00**') AND mpoint('**2020-11-20 19:00:00**')ORDER BY Distance(StartPoint(A.Trip), StartPoint(B.Trip)LIMIT **5**; 数据库优化器 → 可执行语句
实体	Trains,train	Trains
	5, 100, 6: 00, 19: 00	5, 100, 6: 00, 19: 00
操作符	intersects, nearest_neighbor region×region→bool stream×mpoint×int n→stream	Startpoint, Intersection, Distance
索引	UTORdered_rtree	/

图1 自然语言查询转换示例

一致性。在查询生成和优化方面，当引入新查询种类、操作符和索引时，LLM在理解、利用这些操作符和索引并确定其应用上下文方面也面临复杂推理的挑战。因此，需要一套能够评价Text-to-EXE方法以及LLM的基准测试。

问题：现有的Text-to-SQL的基准测试难以用于Text-to-EXE方法的评测。主流基准测试侧重Text-to-SQL评估，无法覆盖移动对象数据库的复杂时空查询需求[6]。早年数据集如Geoquery[7]仅提供自然语言查询。WikiSQL[8]虽给出了自然语言查询和SQL查询对，但仅适用于简单查询。Spider[9]和Bird[10]支持跨域查询，但仅面向固定的数据集，缺乏直接针对移动对象数据库的复杂查询。移动对象数据库查询常为真实场景查询。总的来说，当前基准测试存在双重局限：(1)主流基准测试聚焦Text-to-SQL，缺少面向垂直领域的Text-to-EXE基准测试；(2)现有评估缺少移动对象领域真实场景验证，评价指标强调翻译准确率，较少关注可执行语句查询效率。因此，需要专门的基准测试来捕捉垂直领域的特点和语义，并设计兼顾准确性和效率的复合评价体系。为解决上述问题，构建一个针对移动对象数据库Text-to-EXE的基准测试，需应对以下三个挑战：(1)基于领域知识的基准测试构建方法缺失：移动对象数据库具有复杂的领域知识和多样化的操作符，需要针对查询负载和数据分布使用索引，深入了解领域背景和数据分布并结合实际需求进行设计，以快速生成大量语料；(2)自然语言理解难：时间和空间信息的领域表达和模糊性增加了自然语言理解难度，需处理同义、缩写及歧义等问题，确保用户意图被正确理解；(3)可执行语句生成复杂：因其操作符多样性和查询类型丰富性，生成可执行语句时需考虑实体与操作符组合及查询优化问题。自然语言解析的准确性和查询执行的效率在设计查询时均需兼顾，以确保查询的准确性和高效性。

为此，本文提出了NALMOBench(Natural Language Interfaces for Moving Objects Databases Benchmarking)，一个LLM驱动的移动对象数据库自然语言查询转换复杂真实世界基准测试，是首个能够评估Text-to-EXE系统以及LLM在该领域表现的基准测试。提出的基于LLM的自动化语料库生成方法能够快速生成大量标注的Text-to-EXE语料，具有领域泛化性，并进一步解决自然语言理解和可执行语句生成问题，给出了基线方法。据我们所知，NALMOBench是首个由时空数据库领域专家和交通执法人员合作开发的基准测试。该基准测试支持其他基准测试不曾考虑的包含时空操作的查

询，不仅包含专家设计的在研究中被广泛使用的移动对象查询，如时间段查询、范围查询、最近邻居查询、join查询和轨迹相似性查询，还包括真实交通执法场景查询。在评价指标方面不仅只评价翻译能力，还进一步考量现有基准测试较少考虑的可执行语句执行效率能力。旨在推动Text-to-EXE方法的不断进步，帮助研究人员应对移动对象数据库在实际应用中的复杂挑战。本基准测试填补了移动对象数据库领域Text-to-EXE的空白，并与现有基准测试形成互补关系，共同推动NLIDB系统的全面发展。图2展示了该基准测试系统的架构。主要贡献如下：

(1)提出了NALMOBench，一个用于评估移动对象数据库中Text-to-EXE的新型基准测试平台。包含超过600个自然语言查询与可执行数据库查询对，包含常见移动对象查询以及真实场景查询，按查询类型划分为7类，按难易程度分为3级。

(2)通过结合人工收集的自然语言查询与基于GPT-4o的自动数据增强技术，采用上下文学习和思维链(Chain-of-Thoughts，CoT)方法，正反逻辑结合，构建了一个包含约6000个移动对象查询的高质量语料库。语料库中标注的可执行语句会经过形式化验证。

(3)设计知识检索体系优化自然语言理解算法，分别构建自然语言候选集与数据库候选集，建立分层索引以保证检索效率，实现对移动对象可执行语句的精确解释及关键实体的高效提取。

(4)构建分治法范式优化可执行语句生成方法。以可执行语言模型(Structured Language Model，SLM)为基础构建草图，训练查询类型判断模型。依据识别结果选择子SLM，将相关实体映射至指定槽位并与操作符结合，生成候选可执行语句。随后利用索引枚举与选择机制对候选语句进行后处理。

(5)在构建的基准测试中评估了流行NLIDB系统与LLM。实验结果证明，基线方法指导LLM在可翻译性及翻译精度分别为81.21%和66.14%，翻译效率分数为68.42(相对分数)，优于现有方法。

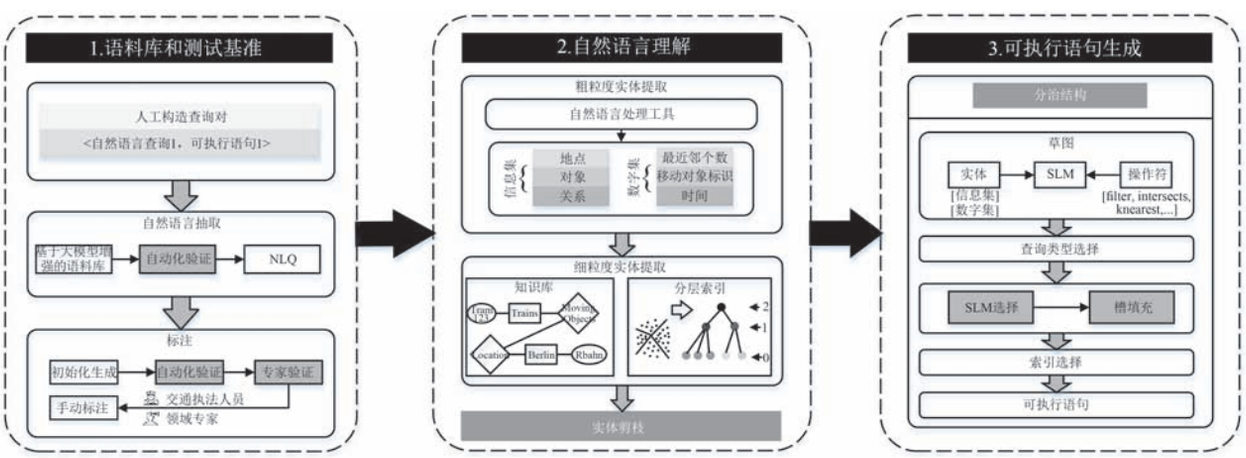


图2　NALMOBench基准架构图

2　相关工作

2.1　移动对象查询

移动对象数据固有的多维时空属性特征对传统数据库管理系统提出了挑战，使数据库领域探索专门的处理机制，以应对连续移动对象数据流的高效管理需求。范围查询旨在检索指定空间范围内满足条件的对象集合。Jensen等人[11]提出了一种使用Hilbert空间填充曲线线性化高维空间数据的方法。最近邻居查询是基于位置服务中的基本操作，需在动态数据集中实时返回与查询点最近的k个对象。Güting等人[12]提出了一套过滤-精炼解决方案，过滤阶段引入动态时空剪枝策略。轨迹相似性查询是从候选集中检索与查询轨迹相似度最高的k条轨迹。许多解决方案也采用过滤-精炼框架，通过利用R树[13]空间索引结构高效缩小搜索空间，加速检索。

2.2　数据库自然语言接口

大量非专业数据库用户的涌现推动了NLIDB的广泛研究[14]。NLIDB在不同的发展阶段有着不同的代表性方法，从基于规则的方法过渡到基于深度神经网络的方法，再到基于预训练语言模型(Pre-trained Language Model，PLM)的方法，然后发展到近年来获得广泛关注的基于LLM的方法。上述发

展有明显的从专有到通用的发展趋势，对于NLIDB泛化性要求在不断提高。早期的NLIDB如NaLIR依赖于手工编写的规则解析自然语言查询。相较于基于规则的方法，基于深度学习的方法如ValueNet能够支持语义更丰富的自然语言查询，提高了跨领域查询的准确性和适应性。PLM的出现如RESDSQL[15]进一步提高了NLIDB的泛化性。近期基于LLM的方法如FinSQL、DIN-SQL[16]、DAIL-SQL[17]、CHESS[18]以及XIYAN-SQL[19]等，能够在较少的人工干预下生成复杂的SQL查询。

2.3 数据库自然语言接口的基准测试

基准测试在NLIDB系统的开发和验证中起着关键作用。早期基于解析的方法如NaLIR、PRECISE[20]和ATHENA++[21]依赖于Geoquery、MAS和FIBEN等数据集，虽在特定领域表现优异但泛化性不足。随着神经网络技术的发展，跨域基准测试逐步兴起。Zhong等人[8]提出了第一个大规模多领域的关系型数据库数据集WikiSQL。WikiSQL包含80 654个自然语言问题和77 840个SQL语句，但其SQL语句仅支持简单操作。为弥补WikiSQL的不足，Yu等人[9]提出了Spider，其数据规模和查询复杂度均得到提升。Spider包含10 181个自然语言问题、5693个SQL语句，涉及多个领域和数据库。尽管WikiSQL和Spider提出了大型跨域数据集，但数据库模式依赖性强，缺乏实际应用场景。KaggleDBQA[22]则通过使用从真实世界数据源中提取的数据集，使系统具备了值链接能力。随着LLM时代的到来，NLIDB的研究迎来新范式[23]。

为验证基于LLM方法的翻译能力，Li等人[10]提出了Bird，包含12 751个自然语言查询与SQL对和95个数据表，涵盖37个专业领域。Bird引入脏数据处理机制，还提出了用于评估Text-to-SQL效率的指标，但其仅关注时间成本。Gkini等人[24]则不仅关注执行时间，还考虑资源消耗。

虽然现有的基准测试能够有效评估当前方法，但NLIDB需要与实际应用对齐。Spider和Bird声称具有高度泛化能力，但由学生专门创建，无法充分代表真实世界的复杂性。ScienceBenchmark[25]是第一个针对科学数据库构建，由专家验证的复杂查询集。Football[26]针对运动这一垂直领域。Spider 2.0[4]针对实际企业级工作流中真实任务。在现实应用中，垂直领域的基准测试仍有研究前景。NLIDB可能需要处理单个或多个相关数据库以满足垂直领域的需求，单个领域的高性能可能比跨领域的泛化能力更为重要。表1总结了NALMOBench和现有部分NLIDB基准测试数据集。

表1 NALMOBench和现有部分NLIDB数据集对比

数据集名称	领域	#查询示例	#数据表
WikiSQL	跨域	80 654	26 521
KaggleDBQA	跨域	272	8
Spider	跨域	10 181	200
Spider 2.0	跨域	632	213
Bird	跨域	12 751	95
Fiben	金融	300	1
ScienceBenchmark	科学	5332	3
NALMOBench	移动对象	600	2

3 问题定义

定义1. 移动对象查询. 本文考虑的查询类型包括时间段查询TI、范围查询RA、最近邻居查询NN、join查询J、轨迹相似性查询TS、异地经营查询CR以及绕路查询DT。这7种查询类型涵盖了移动对象数据库中常见的空间分析、时间分析、轨迹分析以及多条件组合分析需求，能够较为全面地覆盖常见的科研场景和部分现实应用场景，具有较强的代表性。

定义2. 移动对象自然语言查询语料库. 给定自然语言查询集$Q=\{q_1,q_2,\cdots,q_i\}$及其查询种类$Type(q_i)$。其中q_i表示第i个查询，$Type(q_i)$表示q_i的查询类型。移动对象查询语料库Moc定义如下：

$$Moc=\left\{\left(q_i,Type\left(q_i\right)\right)|i=1,2,\cdots,n\right\} \quad (1)$$

定义3. 自然语言理解. 给定自然语言查询Q，自然语言候选集为$NLC=\{nlc_1,nlc_2,\cdots,nlc_m\}$，数据库候选集为$DBC=\{dbc_1,dbc_2,\cdots,dbc_n\}$。给定一个相似性函数$Sim\left(nlc_i,dbc_j\right)$来计算自然语言候选集中的$nlc_i$和数据库候选集中$dbc_j$之间的相似度。对于每一个$nlc_i$，找出与之相似度最高的$dbc_j$。最后，从对齐结果中提取精确的实体$EN$。

定义4. 可执行语句构造. 给定自然语言查询集$Q=\{q_1,q_2,\cdots,q_i\}$，对应的草图S，查询种类$Type(q_i)$，可执行语句模型SLM，操作符集O，槽集$Slots$以及实体映射EN。从自然语言查询集Q中提取草图S。确定q_i的查询类别$Type(q_i)$。根据$Type(q_i)$构建SLM。在模型构建过程中选择合适的运算符O。将自然语言查询中的实体EN映射到SLM中的

槽$Slots$中。结合索引I最终生成可执行查询语句。

定义5. Text-to-executable-language. 指将自然语言查询集$Q=\{q_1,q_2,\cdots,q_i\}$转换为能够从数据库中检索相关数据的可执行语句Y的过程。数据库信息$D=\{LKB, RKB\}$包含地点知识库LKB以及查询关系知识库RKB，其中LKB通常包含数据库中所有地点的信息，RKB封装了与数据库中所有移动对象关系相关的信息，包括唯一标识符、名称和移动对象属性。最终的可执行查询语句需要包含实体信息EN、操作符O以及可选的索引I。最终定义为

$$Y=f\left(Q,D,EN,O,[I]|T\right) \tag{2}$$

其中，函数$f\{\cdot|T\}$表示一个有时间限制T的模型。

4　基准测试构建

现有的基准测试主要面向Text-to-SQL任务，缺乏专门为Text-to-EXE构建的基准测试。尽管现有基准测试声称具备跨域适用性，但未能充分满足特定领域需求，并缺少真实场景查询及效率验证。为此，构建一个面向移动对象数据库的Text-to-EXE基准测试。图3展示了该基准测试构建的工作流程。

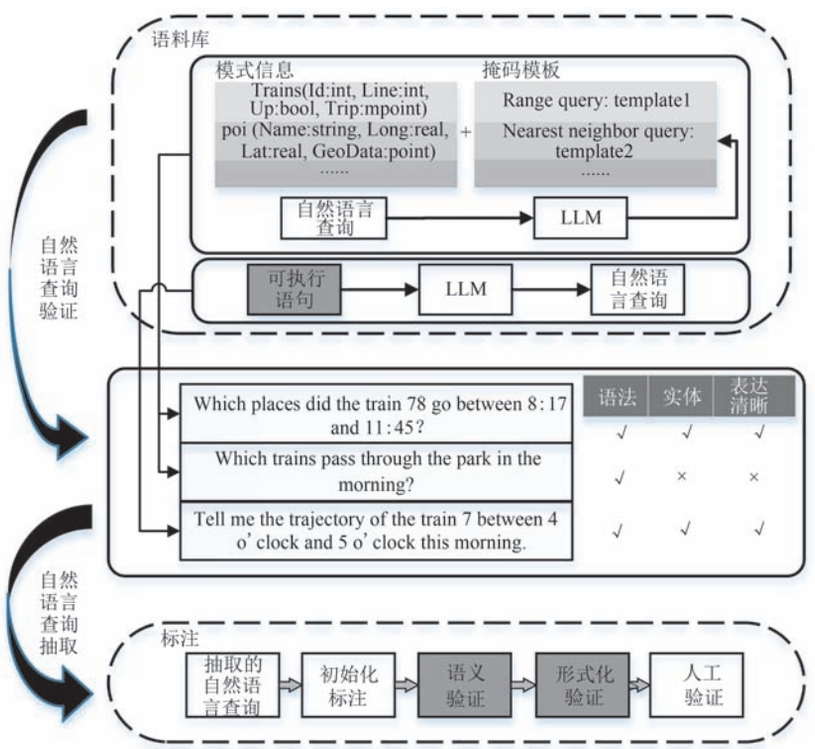


图3　基准测试构建的工作流程

4.1　语料库构建方法

以收集的自然语言查询为基础构建模板，并利用LLM进行数据增强，预先构建了一个包含约6000个移动对象自然语言查询的语料库，涵盖TI、RA、NN、J、TS、CR以及DT查询。

4.1.1　自然语言查询收集

收集并设计了100条鲁棒的移动对象自然语言查询。针对每种类型的查询，从顶级数据库期刊和会议中收集并手动优化了80条自然语言查询，真实场景查询由交通执法人员提出，并经过专家调整。然后，对这100条自然语言查询进行了初步的人工标注，具体标注过程见第4.2.1节。

4.1.2　自动化数据增强

基于收集的查询，为每种类型构建自然语言查询增强模板。基于增强模板，利用GPT-4o进行少样本提示的上下文学习，对模板中的占位符进行数据分布感知的实体替换。对于样本的选择，依据实体类型设置了上下界限定，例如在时间段查询中，先定位查询关系，再依据移动对象确定起始与结束时间后进行填充，使得填充过程既保留了随机生成的多样性，又避免因过于随机而引入大量无效查询。同时，为确保生成样本的多样表达，LLM生成过程中采用高温模式，并在必要时结合基于相似度的过滤策略，以剔除重复或低质量的查询样本。该自动

化流程明确了样本生成的数量与选择规则，为后续步骤提供了丰富的基础语料。

4.1.3 自然语言查询质量验证

为了进一步保证语料库的查询质量，对查询进行二次验证，首先使用自动化的验证工具判断查询是否：(1)语法正确；(2)实体信息与查询种类相对应；(3)不存在模糊信息。筛选出的不符合条件的查询将交由专家进行二次判断。由于在基准测试构建过程中还将进一步进行自然语言查询的提取，初步验证的主要目的是确保查询的语法正确性和鲁棒性，而非与数据库进行交互。

4.2 基准测试构建方法

采用结合人工和自动方法结合的混合策略来构建自然语言查询和数据库查询对。首先，手动创建了100个查询对。随后，利用自动验证方法从现有语料库中提取了约500个自然语言查询。使用NALMO[27]和NALSpatial[28]进行初步标注，并通过语法筛查和形式化验证技术对标注后的查询进行检查。最后，由专家对未标注的查询进行手动标注。

4.2.1 人工生成

邀请了三位时空数据库领域的研究生和两位交通执法一线的执法人员参与查询对的手动构建工作。非真实场景查询由三位研究生完成，异地经营和绕路查询则由执法人员提出问题，研究生根据数据库进行微调，构造相应查询对，并由执法人员验证查询结果。共构建了100条查询对。为确保数据标注的一致性与质量，进行了严格的质量评估。对于非真实场景查询，由三位时空数据库领域的研究生独立完成标注，Fleiss'Kappa系数[29]为0.72，表明标注结果具有较高的一致性。对于真实场景查询，采用执法专家与研究人员协作的模式来进行数据构建，经评估其角色间语义一致性达到85%。此外，所有查询均经过在移动对象数据库中的实际执行结果验证，确认查询结果符合预期，从而进一步保证了整体数据集的准确性和适用性。

与此同时，利用已构建的语料库来自动化抽取自然语言查询。抽取标准包括：(1)确保查询类型对应的模式信息正确，能与数据库信息准确映射；(2)不兼容模式的外部信息需经过验证(如在知识库或现实中存在)；(3)不含模糊信息；(4)不含不支持的查询类型；(5)没有语法错误。最后，利用少样本CoT方法构建提示词，并调用GPT-4o模型完成自动化验证过程，以提取自然语言查询。

4.2.2 自然语言查询标注

利用NALMO提供的方法进行初步标注，但NALMO目前仅支持部分查询类型(如时间段查询、范围查询、最近邻查询、join查询和轨迹相似性查询)的生成，难以直接标注涉及复杂推理的真实场景查询。为了应对复杂查询的生成需求，本文设计了一种基于操作符树的中间表示方法。通过对可执行语句进行语义解析，将其转换为操作符树结构，并进一步泛化为操作符树模板。结合数据库模式信息，将操作符树模板中的抽象节点替换为具体的实体信息，生成不同版本的操作符树实例。随后，利用LLM将这些操作符树实例转化为多样化的自然语言查询表达，从而完成复杂查询的自动化标注过程。为保证自动化标注的质量，进行自动化质量验证，使用手动构建的查询对作为示例，并借助GPT-4o模型将生成的可执行语句与原始自然语言查询进行比较，评估其语义等价性。然后进行Z3形式化验证，解析可执行语句以提取其查询类别、结构和组成部分，为实体创建符号变量，将不同类别子句中的条件转换为Z3约束，检查查询的约束条件是否存在矛盾。对于无法生成或验证失败的查询，将通过专家再次调整或构建。所有真实场景查询均由专家手动构造。

5 自然语言理解

为应对自然语言固有的复杂性挑战，设计了一种两阶段处理方法，包括粗粒度和细粒度自然语言处理模块。初始的粗粒度阶段旨在生成候选实体集。随后，在细粒度阶段，利用预先构建知识库对实体候选集进行剪枝，以提取地点、查询对象、查询关系、最近邻居个数、移动对象标识符和时间信息等。具体过程如算法1所示。

算法1. 实体抽取

输入：n个自然语言查询的列表Q，移动对象知识库MOKB，移动对象信息集MOIS，移动对象数字集MONS

输出：实体EN

1. $doc \leftarrow nlp(Q)$ //利用自然语言处理工具进行粗处理
2. FOR $token \in doc$ DO
3. $MONS \leftarrow Extract\_number(token)$ //构建数字候选集
4. $MOIS \leftarrow Extract\_information(token)$ //构建信息候选集
5. IF *token* 被标签为 *TIME* THEN

6.　　*EN. time* ← *token* //识别时间
7.　IF *token* 表示最近邻居个数 THEN
8.　　*EN. k* ← *token* //识别最近邻居个数
9. FOR *entity* ∈ *MOIS* DO
10.　　*EN. location*, *EN. relation*, *EN. object* ← *Match*(*entity*, *MOKB*) //识别地点，查询关系以及查询对象
11. FOR *number* ∈ *MONS* DO
12.　*number* ← *MIN_SEMANTIC_DIS*(*number*, *EN. object*) //找出与查询对象语义距离最小的数字
13.　*EN. identification* ← *number* //识别查询对象标识
14. RETURN *EN*

对算法1中主要操作的时间复杂度进行如下分析：第1行中对查询列表Q使用自然语言处理工具处理时，文本预处理的复杂度与查询中token的总数有关，记单个查询平均包含L个token，则总体复杂度为$O(n\cdot L)$。在第2至8行，由于对每个token进行实体提取，假设对单个token上的操作可以看作常数时间操作，则这一部分的复杂度为$O(n\cdot L)$。第9至10行中的MOIS遍历过程中，记MOIS中候选实体个数为m。对于每个候选实体，进行数据库对齐，此处若采用暴力搜索，记MOKB中的实体数量为j，复杂度为$O(j)$。若构建索引，匹配操作的平均时间可降低到$O(\log j)$或近似常数时间。因此，该部分的总复杂度为$O(m\cdot f(MOKB))$，其中$f(MOKB)$表示匹配操作的代价。第11至13行对MONS中的数字候选进行遍历，假设候选数字个数为k，并且预先计算语义嵌入或者利用高效相似度计算技术，则每次比较的复杂度降为$O(d)$，其中d为向量维度，可视作常数，则这一部分的复杂度为$O(k)$。算法整体的时间复杂度可以表示为$O(n\cdot L+m\cdot f(MOKB)+k)$。

5.1　自然语言候选实体生成

在此阶段，主要目标是构建自然语言候选集。在自然语言理解过程中，除正确提取实体外，提取效率也是考量因素之一。考虑到自然语言查询语义结构的多样性与复杂性，采用现有主流自然语言处理工具spaCy，以快速近似定位候选实体。利用spaCy进行分词和命名实体识别，构建了两个实体候选集：(1)信息候选集和(2)数字候选集。信息候选集包括地点、查询对象和查询关系等，数字候选集则包含最近邻居的个数、移动对象标识符和时间信息等。粗粒度提取尽可能提取出所有潜在实体信息。首先，通过正则表达式对时间进行准确提取。然后，将与“nearest”等关键字语义距离最近的且存在于数字候选集中的数字认定为最近邻居个数。考虑到查询对象可能的模糊性，需进一步抽取。

5.2　数据库候选实体生成

为解决自然语言实体信息与数据库信息进行对齐问题，提出了一种构建数据库候选集的方法，以实现精确的实体提取。为了准确提取关键实体信息，如地点、对象、关系和查询对象标识，设计了一个移动对象知识库MOKB，包含两个子知识库：(1)关系知识库RKB和(2)地点知识库LKB。

系统会初步分析数据库中的对象，识别包含*mpoint*属性(表示移动对象)的关系，并存入RKB中，这一查询关系中含有可能的移动对象。LKB主要由从数据库中提取的具有点、线和区域属性的对象构成。同时利用GPT-4o来增强LKB，以增加实际存在的同义、缩写和歧义表达。精确的地点信息可从LKB中检索获得。在构建MOKB后，RKB用于确定移动对象的关系和对象。一旦识别出某个对象，通过语义距离找到与该对象最接近且未被识别为其他实体的数字，确定其为对象标识。

为提高知识库检索效率，构建基于MOKB的分层索引结构。分层索引树采用自底向上的构建方法。对于给定的样本数据集，首先使用聚类方法将相似的样本数据聚合为多个簇。随后，通过迭代方式合并最近的一对簇，直至所有簇合并成一个单一簇，此时合并过程结束。该方法最终生成一个多级分层索引树。总构建时间复杂度为$O\left(N+t\cdot C\cdot N+N\cdot L+M+C\log C+N\log(N/C)\right)$，检索的时间复杂度为$O(\log N)$，其中$t$是迭代次数，$C$是城市的数量，$L$是地点的平均长度，$M$是地点的数量。

同时在语义理解过程中需要对查询进行分类，便于后续可执行语句构造。利用构建的语料库进行分类模型训练，将语料库中约6000条自然语言查询按照8∶2的比例划分为训练集和测试集，并选择了TextCNN[30]、LSTM[31]和BERT[32]三种模型。首先，对于任意一条输入查询x，其经过嵌入层和特征提取后得到特征向量表示为$f(x)\in\mathbb{R}^d$。然后使用线性分类器对该特征进行类别预测，其过程可以形式化为：

$$P(y=k|x)=\exp(f(x)\cdot W_k+b_k)/\sum_j\exp(f(x)\cdot W_j+b_j) \tag{3}$$

其中，$W_k\in\mathbb{R}^d$和b_k为类别k的权重和偏置参数，y表

示查询类别训练过程中，我们最小化交叉熵损失函数L，其表达式为

$$L=-\sum_{i=1}^{N}\sum_{k=1}^{K}y_{i,k}\log P(y=k|x_i) \quad (4)$$

其中，N为样本总数，K为类别数，$y_{i,k}$为标签。LSTM和BERT在测试集上的预测准确性相当，且均优于TextCNN。由于BERT模型参数量庞大，其训练时间约为1.5分钟，而LSTM的训练时间仅需约1秒。TextCNN在分类任务中表现出色，但无法处理更复杂的任务。深度学习模型如BERT虽然具有强大的推理能力，但其复杂度过高。LSTM在小规模数据场景下，能更好权衡模型复杂度与任务性能。

6 可执行语句生成

结构化语句包含SQL以及垂直领域的可执行语句等。本文中均特指移动对象数据库支持的可执行语句。多种类移动对象查询的复杂性使得难以开发一个通用SLM。通过枚举为每个查询单独构建SLM并进行实体和操作符的组合代价高昂，尤其在现实应用场景中更为明显，可见后续证明。因此，提出了一种分治策略，以减少模型设计成本，并构建了一个查询优化模型通过索引选择优化生成的可执行语句。本文中涉及的表达式分为对象声明表达式和查询表达式。具体可执行语句构造流程如图4所示。为深入理解该构造过程的复杂性，下文将形式化地给出其复杂度的理论分析与证明：

输入参数关系集合$R=\{r_1,r_2,\cdots,r_n\}$，实体集合$\varepsilon=\{e_1,e_2,\cdots,e_m\}$，索引集$I=\{in_1,in_2,\cdots,in_k\}$，操作符集$O=\{o_1,o_2,\cdots,o_l\}$，工作负载$W=\{q_1,q_2,\cdots,q_t\}$，其中$q_j$的频率为$f_j$，且$q_j$访问实体$\varepsilon_j\subseteq\varepsilon$，约束条件$C$，例如CPU资源和运行时间等。选择子集$I^*\subseteq I$、$O^*\subseteq O$、$\varepsilon^*\subseteq\varepsilon$，最小化总执行成本：

$$TotalCost=\sum_{j=1}^{t}f_j\cdot cost(q_j,O^*,I^*,\varepsilon^*) \quad (5)$$

给定阈值TH，是否存在$I^*\subseteq I$、$O^*\subseteq O$、$\varepsilon^*\subseteq\varepsilon$，使得

$$TotalCost(O^*,I^*,\varepsilon^*)\leqslant TH \quad (6)$$

且满足约束C：$|I^*|\leqslant p_1$，$|O^*|\leqslant p_2$，$|\varepsilon^*|\leqslant p_3$。

接下来归约到索引选择问题（Optimum Index Selection Problem，OISP）。已知OISP属于NP-Hard问题的范畴[33]，当扩展归约至包含操作符和实体，若任一子问题（如索引选择）是NP-Hard，且其余部分未引入结构性约束，则整个问题可被视为具有较高计算复杂度。

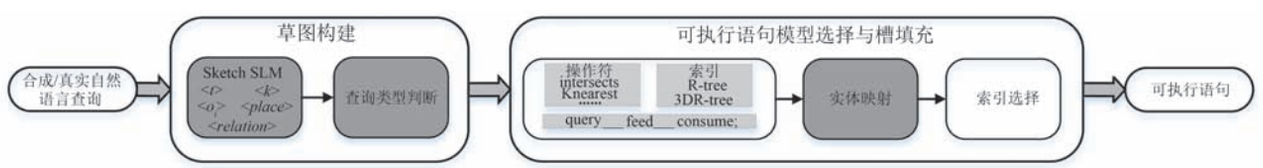


图4 可执行语句构造流程

6.1 可执行语句模型草图

图4展示了草图SLM，其中〈k〉、〈t〉、〈o_i〉、〈*place*〉和〈*relation*〉分别表示待填充的最近邻居个数、时间、查询对象及其标识、地点以及移动对象关系。如果地点被封装在查询关系中，〈*place*〉占位符则将替换为下面表达式，其中〈*tmp relation*〉和〈*name*〉分别表示移动对象关系和对象名称。

query 〈*tmp relation*〉 feed filter [. Name = "〈*name*〉"] extract [Trip]）；表2列出了部分操作符，包括名称、语法及其含义。

6.2 类型选择

在自然语言理解过程构建了移动对象自然语言查询类别分类器，然后利用该分类器获取输入的自然语言查询种类，在可执行语句模型的草图的基础上会根据查询类别的不同进行子可执行语句模型模块的选择和组合，从而完成最终与查询类别相对应

表2 SECONDO中的操作符

操作符	语法	含义
filter	stream×filter condition→stream	流元素谓词过滤
intersects	{mpoint, line, region}×{mpoint, line, region}→bool	空间相交性判定
shortestpathlf	orel(tuple())×IDENT×point2→stream(tuple())	最短路径计算
knearest	stream×mpoint×int n→stream	k近邻流计算
dist_euclidean	pointseq×pointseq→real	轨迹相似度计算

的子可执行语句模型的选择，减小候选可执行语句构造过程中的搜索空间。

6.3　可执行语句模型选择与槽填充

针对不同类型查询，依据预定义映射规则将相应实体映射到对应槽位，并结合所需的操作符生成最终的可执行语句。可执行语句模型如附录1中表5所示，此处仅给出每种查询的可执行语句模型示例，根据是否包含具体对象、时间以及地点等信息并结合所需的操作符，模型均有不同变体。以真实场景查询为例进行解释，绕路行为检测通过*getstartpoint*和*getendpoint*操作符获得车辆的起始点和终点，并利用*shortestpathlf*操作符构建最短线路。利用*dist_euclidean*操作符判断查询预期线路与实际路线的轨迹相似度是否满足设定的阈值。

上述分治框架通过查询类型识别以及预定义的规则结构将可执行语句构造拆解为多个子问题，显著缩小LLM在生成可执行语句时的搜索空间，从而提升其准确率与稳定性。以检索增强生成（Retrieval-Augmented Generation，RAG）框架为例，本文在自然语言理解和可执行语句模型的基础上引入RAG，通过检索器动态检索知识库以获取实体信息。接下来，借助少样本示例的上下文学习方法，采用CoT推理机制，将生成过程分解为实体信息选择、可执行语句模型匹配以及实体映射三个子问题，利用GPT-4o多路径生成输出多个候选解，使用执行结果的自一致性投票机制选出最优候选，并通过后续的后处理阶段实现查询效率直接进行优化，生成最优查询。

在后处理阶段，聚焦于索引的使用。首先，计算和收集每个时空关系中涉及的记录的数量。然后，根据操作符约束和查询中涉及的关系来估计过滤率，其中过滤率是指满足特定查询条件的记录相对于数据集中记录总数的百分比。对于具有高过滤率的查询，以生成的可执行语句为基础，利用索引来枚举候选可执行数据库查询。随后，使用了一个成本模型，利用采样的小规模数据来执行可执行语句，以预测每个候选可执行数据库查询的时间开销，最终为用户提供最优的可执行查询语句。此外，已构建在线用户交互界面，实现了初步的用户反馈机制。在现阶段，系统通过检测查询失败的原因（例如实体不正确、查询类别不匹配或查询结果为空等）并提供直观的用户示例，允许用户调整后重新查询。同时，针对查询中涉及的实体问题，允许用户提交反馈，进而将缺失的实体信息补充到知识库中。未来，在此基础上进一步引入允许用户直接对查询结果进行评价和打分的机制，并设计基于已有规则的反馈模型，以指导LLM在多路径查询生成时根据用户反馈而非单纯的执行结果来自动调整并选取用户最满意的查询结果，从而不断提升系统的准确性和效率。

7　实验结果与分析

实验平台配置如下：Intel（R）Core（TM）i9-11950H CPU、2.60 GHz、32 GB内存、512 GB硬盘，操作系统为Ubuntu 22.04（64位，内核版本6.8.0-49-generic）。旨在研究现有Text-to-SQL方法以及LLM在Text-to-EXE问题中的表现，并探讨这些方法在移动对象数据库垂直领域中的适用性和鲁棒性。实验数据截至2024.12.31。为全面评估，不仅关注查询转换的准确性，还考察转换效率和执行效率。转换效率指从自然语言查询到可执行语句的转换速度。执行效率指的是生成的可执行语句在数据库中的执行效率。数据集和基准测试模型代码见https://github.com/zhongmove/NALMOBench。

7.1　实验设置

7.1.1　查询分类及难度标准

如第4节所述，利用人工和LLM标注相结合的方法进行测试用例的构建。尽管利用LLM进行生成可以减少工作量，但仍然需要一定程度的手动审查，因此最终标记了600对自然语言查询与可执行查询对，能够覆盖现有流行的移动对象查询种类。现有的查询主要有两种，一种是面向数据库而专门设计，一种是面向真实需求而设计，本文则都进行了考虑。专有设计的查询包含TI、RA、NN、TR和J等查询，面向真实场景的查询包括CR及DT等查询。除此之外，Spider的SQL查询难度基于查询长度，分为简单、中等、困难和极难四个难度。在此基础上结合移动对象查询的复杂查询类型对查询进行切分，分为简单、中等和困难，其中TI查询仅涉及时间、查询关系和查询对象等实体，为简单难度；RA、NN、J和TS查询涉及更加复杂的实体以及操作符，为中等难度；CR和DT查询由于涉及复杂的真实场景，可能存在复杂的多步查询，为困难难度。这种划分确保了从自然语言角度和可执行语句角度进行多样化采样。自然语言查询和可执行语句对示例见附录1中表6。

7.1.2 数据集

实验中使用的数据集包括真实世界营运车辆运动轨迹nanjingtest和合成的火车轨迹berlintest。其中真实采集的移动轨迹为2024年10月22日0时至24时期间南京252辆出租车、公交车和汽车的轨迹片段，涉及移动对象点数量为714938个；火车移动轨迹为2003年11月20日6时至9时期间柏林562辆火车的轨迹片段，涉及移动对象点数量为51544个火车轨迹数据来源于2003年的采集与合成，由于合成后数据库模式信息统一存储为2020年，后续实验与示例均以2020年作为时间基准。nanjingtest同时存储了南京市的行政区、部分道路和地点信息，有9900个地理位置。地理位置由9000个点、887条线和13个区域组成；berlintest存储了柏林的城市地理信息，包括POI和河流，有8078个地理位置。地理位置由3040个点、4708条线和330个区域组成。具体信息见表3。

表3 数据集信息

数据集	#表格	#点	#线	#区域	#移动对象	时间范围
nanjingtest	6	9000	887	13	252	[2024.06.15:00:00:00, 2024.06.15:24:00:00]
berlintest	50	3040	4078	330	562	[2003.11.20:06:00:00, 2003.11.20:09:00:00]

数据集考虑了数据来源的科学性与扩展基础，涵盖真实与合成轨迹数据，并覆盖城市交通与铁路运输两类典型移动场景。nanjingtest与berlintest分别提供真实轨迹与模拟轨迹，具备空间尺度广、时间跨度完整及对象类型多样的特点，能够支撑模型在不同语义环境下进行泛化评估。同时，数据集设计覆盖七类关键时空查询类型，增强了对移动对象数据库操作符的表达能力。因此，当前两个数据集在代表性与复杂性方面已具备支持Text-to-EXE任务研究的能力，并为未来持续扩展奠定良好基础。

7.2 评价指标

在Text-to-SQL领域，系统的评估主要有三个指标，分别是精确匹配(Exact Match，EM)、执行准确率(Execution Accuracy，EX)和有效效率评分(Valid Efficiency Score，VES)。EM用于衡量模型生成的SQL与标注SQL之间的相似程度。EX计算数据集中正确执行的SQL结果占比。VES旨在评估模型生成的有效SQL的执行效率，即SQL的执行时间。为了评估Text-to-EXE系统的自然语言理解能力，提出了一种新的度量标准可翻译性(Translatability，TA)。生成时间也极大影响用户体验，但基于LLM的方法主要时间花费在推理过程，而非基于LLM的方法主要时间消耗在实体抽取过程中，无法统一比较，所以会在后续单独讨论。给定系统生成的可执行数据库查询集合EQ和输入的自然语言查询集合N，TA定义为

$$TA=\frac{|EQ|}{|N|} \tag{7}$$

与EM不同，精确匹配可能会忽视那些在逻辑上正确但语法表达不同的查询，而可翻译性通过考虑所有结构上可行的数据库查询来扩大评估范围。

为了评估可执行语句的生成能力，提出了翻译精确率(Translation Precision，TP)。该指标由与预期结果相符的可执行语句集和自然语言查询集合N的比值定义。对生成的可执行语句和标准可执行语句的执行结果进行比较，假定拥有相同的执行结果则表示等价。同时，将执行语句在给定的切分小规模数据集上执行。令S_n表示第n个样例的可执行数据库查询集合，G_n表示第n个样例的标准数据库查询集合，$R(S_n)$表示执行S_n后的结果集合，$R(G_n)$表示执行G_n后的结果集合。TP定义为

$$TP=\frac{1}{N}\sum_{n-1}^{N}\Big(1(S_n,G_n)\cdot 1\big(R(S_n),R(G_n)\big)\Big) \tag{8}$$

$1(A,B)$是一个指标函数，定义为

$$1(A,B)=\begin{cases}1, \text{if } A=B\\ 0, \text{otherwise}\end{cases} \tag{9}$$

为了全面评估生成的可执行语句的质量，构建考虑效率指标翻译效率评分(Translation Efficiency Score，TES)，包括执行时间和综合度量中的CPU时间。与VES不同，TES不仅考虑了可执行语句运行的持续时间，还加入了CPU时间，从而提供了一个更全面的评估维度。令$E(\cdot)$表示执行时间，$C(\cdot)$表示CPU时间。这两个函数用于在特定环境下的绝对时间，TES定义为

$$TES=\frac{1}{N}\sum_{n-1}^{N}\Big(1\big(R(S_n),R(G_n)\big)\cdot QE\Big(Y_n,\hat{Y}_n\Big)\Big) \tag{10}$$

$$QE\Big(Y_n,\hat{Y}_n\Big)=\sqrt{\frac{E(Y_n)}{E\big(\hat{Y}_n\big)}}\Big/\sqrt{\frac{C(Y_n)}{C\big(\hat{Y}_n\big)}} \tag{11}$$

其中，$QE(\cdot)$表示生成的可执行语句相对于标准可执行语句的相对效率，Y_n为标准可执行语句，$\hat{Y}_n$为

生成的可执行语句。引入平方根函数有助于减少极端情况对整体评估结果的干扰。

7.3　实验对比方法

鉴于Text-to-SQL技术的快速发展，从3个主要类别中选择了8个具有代表性的方法：基于规则的方法、基于神经网络的方法和基于LLM的方法。尽管这些类别的流行程度随时间变化，但本文认为所选择的代表性方法在基本原理上具有显著的共性，这些共性可能相互重叠或包含。虽然基于LLM的方法最近已成为主导范式，但这并不能表明基于规则和基于神经网络的方法在Text-to-EXE任务的特定领域中缺乏适用性。因此，对这3类方法进行了全面实验，以获得整体评估：(1)基于规则的方法：本文采用NaLIR和ATHENA++进行对比。NaLIR利用解析树进行实体抽取和对齐，并根据用户反馈构建SQL。ATHENA++通过两阶段生成方法使用OQL查询作为中间表示；(2)基于神经网络的方法：本文使用ValueNet作为对比方法，该方法扩展了SemQL语法。ValueNet采用BART作为编码器，且开源可用，但严重依赖训练数据；(3)基于LLM的方法：选取GPT-4o和开源模型LLaMA3作为基座模型进行对比。在Bird榜单中，前十名方法中超过半数基于GPT-4o。同时，LLaMA3被视为最广泛使用的开源模型之一。此外，选取基于微调的XiYan-SQL、基于RAG框架的CHESS以及基于智能体的OpenSearch-SQL进行对比。

本文提出了NALMO+进行对比。NALMO+是集成到SECONDO系统中的混合架构代数模块，并具有支持自然语言查询的操作符。此外，为进一步验证本文方法对于LLM的Text-to-EXE智能的提升，在自然语言理解和可执行语句模型的基础上以RAG框架为例，结合GPT-4o进行对比。

7.4　实验结果分析

7.4.1　实验结果

首先考虑这些方法的查询转换能力。表4展示了所有方法的TA和TP结果，并详细给出了不同难度下以及不同查询类别的数据。由于移动对象数据库的可执行语句与标准SQL在语法结构和语义表达上存在差异，基于规则的方法仅考虑自然语言理解过程。NaLIR将构建的语义解析树的节点与SQL语句进行映射，在面对移动对象查询时对于实体的识别以及操作符的选择错误率高，TA表现不佳，且TP无法直接计算。ATHENA++结合领域本体知识与自然语言处理工具CoreNLP在面对自然语言理解的过程中能够对于涉及实体数量少且不复杂的查询能够进行有效的理解，但是涉及模糊实体如同义、缩写和歧义等则无法进行有效提取和映射。尤其对于真实场景查询，ATHENA++则无法处理。

由于ValueNet使用中间表示层作为输出，因此指定的后处理步骤对于将中间表示转换为SQL查询至关重要。由于会使用数据库内容作为输入，在TA上能够达到和使用自然语言处理工具相当的结

表4　可翻译性和翻译精度结果

方法	TA/%								TP/%							
	简单	中等				难		全部	简单	中等				难		全部
	TI	RA	NN	J	TS	CR	DT		T	RA	NN	J	TS	CR	DT	
NaLIR	5.84	3.06	1.79	1.05	2.00	0.00	0.00	2.52	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ATHENA++	37.27	31.63	21.43	27.37	28.00	10.17	23.53	29.87	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ValueNet	35.04	20.41	19.64	18.00	16.95	11.86	15.69	24.33	16.79	14.29	10.71	5.26	2.00	3.39	5.88	9.23
LLaMA3-70B(0-shot)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LLaMA3-70B(1-shot)	47.45	30.61	24.62	20.00	15.00	5.08	29.41	27.35	21.17	17.35	16.07	8.42	2.00	1.69	3.92	11.41
LLaMA3-70B(5-shots)	54.75	35.71	35.71	24.21	18.00	15.25	33.33	33.05	25.55	21.43	25.00	11.58	9.00	10.17	9.80	16.95
GPT-4o(0-shot)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GPT-4o(1-shot)	56.20	40.82	39.29	29.47	19.00	6.78	35.29	34.90	23.36	20.41	19.64	9.47	3.00	1.69	11.76	13.76
GPT-4o(5-shots)	81.75	48.98	51.79	31.58	52.00	16.95	39.22	50.84	71.53	47.96	51.79	31.58	16.00	13.56	23.53	40.27
NALMO+	89.05	79.78	78.57	73.68	77.00	45.76	58.82	68.96	72.99	61.22	57.14	61.05	65.00	37.29	41.18	60.07
XiYan-SQL	84.67	74.49	73.21	70.53	76.67	68.41	65.36	73.33	69.34	40.82	48.21	21.05	13.67	13.56	11.76	31.20
CHESS	86.13	76.53	80.35	71.58	78.33	69.49	58.82	74.46	72.12	51.02	55.36	34.73	60.00	33.89	35.29	48.92
OpenSearch-SQL	86.86	78.57	76.79	73.68	78.00	71.19	70.59	76.53	71.90	53.06	51.79	32.63	64.67	35.60	37.25	49.56
NALMO++GPT-4o	91.24	81.63	80.36	83.16	84.67	72.88	74.51	**81.21**	87.59	71.43	71.43	66.31	68.33	50.85	47.05	**66.14**

注：TI是时间段查询；RA是范围查询；NN是最近邻查询；J是join查询；TS是轨迹相似性查询；CR是异地经营；DT是绕路。

果。ValueNet依赖于大规模的数据训练，使用了约300个训练样本，但生成可执行语句的能力有限，所有类别的TP均低于20%，且难度越高TP越低。

为进一步分析LLM的性能，主要使用零样本和少样本提示。由于移动对象数据库可执行语句与标准SQL存在差异，在没有样本提示的情况下，LLM无法给出所需数据库的可执行语句。GPT-4o在少样本情况下的泛化能力更强，TA和TP相比于LLaMA3分别提升了约50%以上和100%以上。由于移动对象可执行语句的语义表达丰富，且没有固定的操作符和实体组合，单样本提示使得LLM难以解决未见过的查询。相比于单样本，在提升至5样本时，LLaMA3的TA和TP均得到了20%以上的提升，GPT-4o则均得到了40%以上的提升。但当样本量继续扩大则可能会加剧幻觉问题，产生操作符的乱用，甚至构造不存在的操作符。

XiYan-SQL尽管在传统SQL翻译任务上表现优异，但在面对结构复杂、语义动态变化的移动对象查询时，即使通过微调也会生成更倾向于声明式SQL而非所需的命令式可执行语句。即便在少样本提示下引入可执行语句样例，其TP仍较低，说明其对目标语义的理解存在偏差。CHESS基于RAG与智能体框架的方法，利用外部知识库辅助生成过程，在少样本设置下能够生成符合可执行语句结构的结果。然而，由于其对时空操作符识别能力较弱，生成语句中常出现错误运算符，导致执行失败。虽然TA较高，但TP受限于领域理解与提示词的设计。OpenSearch-SQL采用智能体提取模式信息，并结合CoT与投票机制生成候选查询路径。多路径生成策略提高了生成多样性，使其在TA指标上表现良好。但由于缺乏对移动对象数据库语义的深入建模，大量生成语句因操作符使用错误而无法执行，导致TP指标降低。此外，推理时间显著增加，影响实际部署效率。一方面，声明式SQL训练目标与命令式可执行语句输出之间的结构性鸿沟限制了模型迁移能力；另一方面，时空操作符语义建模不足及领域知识匮乏成为TP提升的关键瓶颈。

图5展示了不同方法的TES。基于可执行语句的白盒特性，可以对于查询的效率进行直接控制。小模型直接和训练数据挂钩。LLM也没有接触到与数据库索引直接相关的足够多的训练数据，尤其是如何优化查询以利用索引。需要大量数据重新进行训练或者微调。如果LLM未能准确把握数据的分布情况，它可能无法做出合适的索引选择。OpenSearch-SQL、XiYan-SQL以及CHESS均难以理解可执行语句的语义结构。但除基于微调的方法外，引入本文的可执行语句优化规则能取得接近于基于优化器优化后的TES。

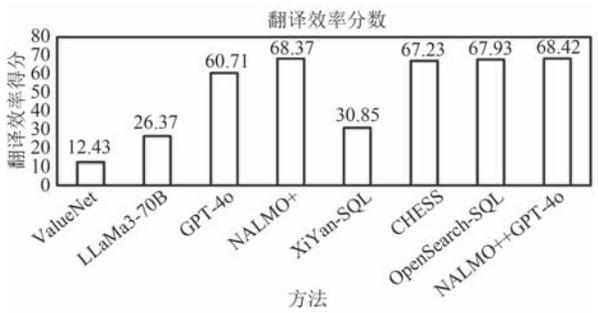


图5 翻译效率分数结果

总之，这些结果表明，NALMOBench是一个极具挑战性的基准测试，也因此证明了Text-to-EXE仍处于初始阶段，有很大的研究空间。

7.4.2 查询复杂性以及查询特性影响

查询的复杂度主要受查询难度的影响。对于简单查询TI，主要只涉及时间、查询对象以及查询关系等信息，可执行语句形式相对固定，仅涉及时间操作符，相对简单，各方法在TA和TP上的表现均为最佳。对于中等难度的RA、NN、J和TS来说，涉及的实体信息更加丰富，包含时间、地点、查询关系、查询对象、最近邻居标识和最近邻居个数等，增加了自然语言理解的难度。同时在可执行语句构造过程中所涉及的操作符进一步增加，操作符和实体的组合的复杂度也随之上升。这里主要探讨LLM存在的问题：(1)实体映射存在问题，涉及日期的变换，模糊查询地点等则无法准确识别；(2)操作符的选择问题，面对未见过的操作符则无法使用，甚至会虚构操作符；(3)缺少部分查询模块，有时会缺少部分所需可执行的模块，导致可执行语句无法运行。在困难查询中，CR和DT查询通常涉及多段查询或嵌套查询。由于实际查询往往需要多次查询处理，现有方法在这些情况下存在挑战。

NALMO+指导LLM的框架虽然优于现有技术，在处理复杂查询时仍有提升空间，但同时说明本文基准测试具有挑战性。例如，在真实场景中的异地经营查询和绕路查询可能涉及多个子查询的组合与优化。这些复杂查询不仅需要高效的语句生成能力，还需要对可执行语句进行精确控制(如操作符选择和索引策略)。然而，现有的LLM和通用方法在这一领域表现不佳。以CHESS为例，虽

然其在Bird数据集上取得了71.1%的执行准确率，但在Spider 2.0数据集中面对复杂查询时，准确率仅达到1.28%。这表明即使在SQL领域复杂查询的处理就存在较高的难度，而可执行语句（计划）相比于SQL更加复杂且难以处理。为进一步提升现有方法对于复杂查询的处理能力，后续优化已有的移动对象数据库知识和规则来指导LLM进行生成。

7.4.3　生成时间分析

图6展示了不同系统生成可执行语句的时间。基于规则的方法仅考虑自然语言理解过程。NaLIR的生成时间主要受自然语言查询的长度和涉及实体的数量影响。在10个token以内的查询平均时间为4.14秒，10个以上为70.75秒。ATHENA++在结合领域本体知识与自然语言处理工具CoreNLP的基础上进行自然语言理解，平均时间为7.74秒。

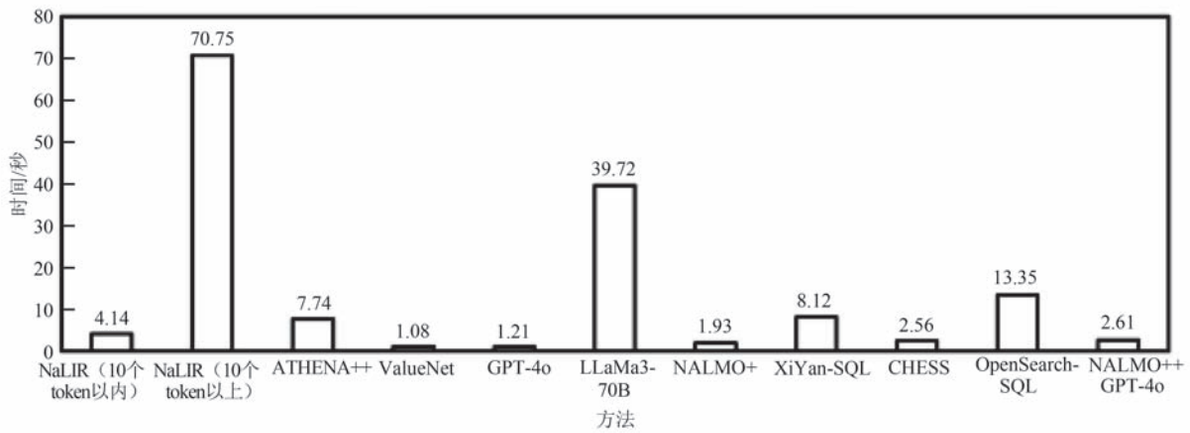


图6　生成时间

对于基于具有148M参数的小型语言模型ValueNet，每个查询的平均推理时间约为1秒。对于GPT-4o，每个查询的平均推理时间为1.21秒。然而，GPT-4o运行在OpenAI控制的云端大型硬件基础设施上，可能存在响应时间过长的情况。LLaMA3-70B的每个查询的平均响应时间为39.72秒。对于像XiYan-SQL基于微调的方法在3B模型上也需要约2小时的微调时间，微调后的可执行语句生成时间约为8秒。CHESS查询响应时间约2.5秒。OpenSearch-SQL查询响应时间约13秒。NAMLO+整体转换时间为2.5秒～3秒，对于其在TA、TP和TES上的表现来说是可接受的。通过利用训练好的模型进行查询类别的快速定位，可以迅速缩小可执行语句构造的搜索空间。

7.4.4　消融实验

本文进行了系统的消融实验，以进一步验证所提出自然语言理解框架（Natural Language Understanding，NLU）中关键模块的有效性，实验结果如图7所示。通过构建三种模型变体（NLU-KB、NLU-Corpus、NLU-KB-Corpus），分别去除知识库模块、语料库模块及其组合，并评估其在实体信息提取率（Entity Information Extraction Rate，EI）与查询类型识别率（Query Type Recognition Rate，QT）上性能变化。实验结果表明，知识库模块显著提升了语义理解与多层级信息抽取能力，缺失时EI下降至25.15%，语料库模块的移除则导致QT下降至15.33%，凸显其在上下文建模与意图识别中的关键作用。二者协同有效增强了系统对复杂自然语言查询的理解精度与适应能力。

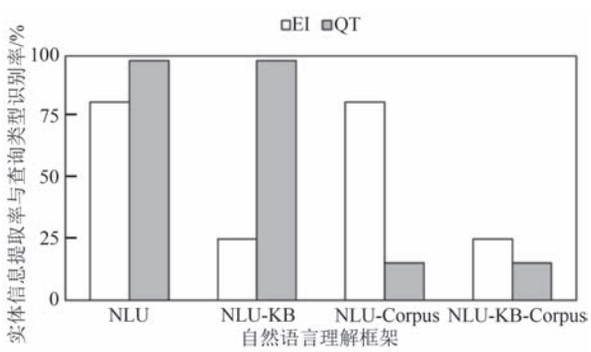


图7　消融实验

8　总　结

本文提出了一个新基准测试NALMOBench，旨在评估自然语言查询到可执行语句转换的能力与质量，涵盖了合成移动对象数据以及真实移动对

象数据。本文给出了为移动对象数据库这一垂直领域的基于LLM的数据库语料生成方法，包含了常见的移动对象查询以及复杂的真实场景查询。本文将Text-to-EXE问题拆解为自然语言理解和可执行语句构造问题，并给出了基线方法。实验结果表明，NALMOBench对现有Text-to-EXE方法以及LLM提出了显著挑战，可以作为评估移动对象数据库领域Text-to-EXE系统的新基准测试。此外，尽管本文以移动对象数据库为核心探讨Text-to-EXE问题，但所提出的基准测试构建与查询转换框架充分体现了领域泛化性。具体来说，通过构建适用于不同领域的自然语言查询模板，并结合数据分布感知的填充方法和LLM的生成能力，可以生成符合空间、时序[34]、图数据库等领域需求的查询语料。在自然语言理解、知识库构建和可执行语句构造过程中，系统预留了灵活扩展的接口，使得不同领域的实体提取和可执行语句模型构造成为可能。

参 考 文 献

[1] Li F, Jagadish H V. Constructing an interactive natural language interface for relational databases. Proceedings of VLDB Endow., 2014, 8(1): 73-84

[2] Brunner U, Stockinger K. ValueNet: A natural language-to-SQL system that learns from database information//Proceedings of the 37th IEEE International Conference on Data Engineering (ICDE). Chania, Greece, 2021: 2177 - 2182

[3] Zhang C, Mao Y, Fan Y, et al. Finsql: Model-agnostic llms-based text-to-sql framework for financial analysis//Companion of the 2024 International Conference on Management of Data (SIGMOD/PODS). Santiago, Chile. 2024: 93-105

[4] Lei F, Chen J, Ye Y, et al. Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. arXiv preprint, 2024, arXiv: 2411.07763

[5] Güting RH, Behr T, Düntgen C. SECONDO: A platform for moving objects database research and for publishing and integrating research implementations. IEEE Data Eng. Bull., 2010, 33(2): 56-63

[6] Zhao Meng, Chen Ke, Shou Li-Dan, et al. Converting complex natural language query to sql based on tree representation model. Journal of Software, 2022, 33(12): 4727-4745 (in Chinese)
(赵猛，陈珂，寿黎但等. 基于树状模型的复杂自然语言查询转SQL技术研究. 软件学报，2022，33(12):4727-4745)

[7] Zelle J M, Mooney R J. Learning to parse database queries using inductive logic programming//Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI). Portland, USA, 1996: 1050-1055

[8] Zhong V, Xiong C, Socher R. Seq2sql: Generating structured queries from natural language using reinforcement learning. arXiv preprint, 2017, arXiv: 1709.00103

[9] 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//Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP). Brussels, Belgium, 2018: 3911-3921

[10] 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//Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems (NeurIPS). New Orleans, USA. 2023

[11] Jensen C S, Lin D, Ooi B C. Query and update efficient b+-tree based indexing of moving objects//Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB). Toronto, Canada, 2004: 768-779

[12] Güting R H, Behr T, Xu J. Efficient k-nearest neighbor search on moving object trajectories. The VLDB Journal, 2010, 19(5): 687-714

[13] Cheung K L, Fu A W. Enhanced nearest neighbour search on the r-tree. SIGMOD Rec., 1998, 27(3): 16-21

[14] Li Guo-Liang, Zhou Xuan-He, Sun Ji, et al. A survey of machine learning based database techniques. Chinese Journal of Computers, 2020, 43(11): 2019-2049 (in Chinese)
(李国良，周煊赫，孙佶等. 基于机器学习的数据库技术综述. 计算机学报，2020，43(11):2019-2049)

[15] Li H, Zhang J, Li C, et al. RESDSQL: decoupling schema linking and skeleton parsing for text-to-sql//Thirty-Seventh AAAI Conference on Artificial Intelligence, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, Thirteenth Symposium on Educational Advances in Artificial Intelligence (AAAI). Washington, USA, 2023: 13067-13075

[16] Pourreza M, Zhang R, Yang K, et al. DIN-SQL: Decomposed in-context learning of text-to-sql with self-correction//Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems (NeurIPS). New Orleans, USA, 2023: 36339-36348

[17] Gao D, Wang H, Li Y, et al. Text-to-SQL empowered by large language models: A benchmark evaluation. Proceedings of VLDB Endow ment, 2024, 17(5): 1132-1145

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

[19] Gao Y, Liu Y, Li X, et al. Xiyan-sql: A multi-generator ensemble framework for text-to-sql. arXiv preprint, 2024, arXiv: 2411.08599

[20] Popescu A, Etzioni O, Kautz H A. Towards a theory of natural language interfaces to databases//Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI). Miami, USA. 2003: 149-157

[21] Sen J, Lei C, Quamar A, et al. ATHENA++: natural language querying for complex nested SQL queries. Proceedings

of VLDB Endow., 2020, 13(11): 2747-2759

[22] Lee C, Polozov O, Richardson M. Kaggledbqa: Realistic evaluation of text-to-sql parsers//Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP). Virtual, 2021: 2261-2273

[23] Mitsopoulou A, Koutrika G. Analysis of text-to-SQL benchmarks: Limitations, challenges and opportunities//Proceedings 28th International Conference on Extending Database Technology (EDBT). Barcelona, Spain, 2025: 199-212

[24] Gkini O, Belmpas T, Koutrika G, et al. An in-depth benchmarking of text-to-sql systems//Proceedings of International Conference on Management of Data (SIGMOD). Virtual, 2021: 632-644

[25] Zhang Y, Kosten C, Katsogiannis-Meimarakis G, et al. Sciencebenchmark: A complex real-world benchmark for evaluating natural language to SQL systems. Proceedings of VLDB Endow ment, 2023, 17(4): 685-698

[26] Fürst J, Deriu J, Nooralahzadeh F, et al. Evaluating the data model robustness of text-to-SQL systems based on real user queries//Proceedings of the 28th International Conference on Extending Database Technology (EDBT). Barcelona, Spain, 2025: 158-170

[27] Wang X, Liu M, Xu J, et al. NALMO: Transforming queries in natural language for moving objects. GeoInformatica, 2023, 27(3): 427-460

[28] Liu M, Wang X, Xu J, et al. NALSpatial: A natural language interface for spatial databases. IEEE Transactions on Knowledge and Data Engineering, 2025, 37(4): 2056-2070

[29] Fleiss J. Measuring nominal scale agreement among many raters. Psychological Bulletin, 1971, 76(5): 378

[30] Kim Y. Convolutional neural networks for sentence classification//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar, 2014: 1746-1751

[31] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput., 1997, 9(8): 1735-1780

[32] Devlin J, Chang M, Lee K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding// Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). Minneapolis, USA, 2019: 4171-4186

[33] Piatetsky-Shapiro G. The optimal selection of secondary indices is NP-Complete. SIGMOD Rec, 1983, 13(2): 72-75

[34] Lin Y, Xu J, Wang X, et al. NLITS: A natural language interface for time series databases.//Web and Big Data - 8th International Joint Conference (APWeb-WAIM). Jinhua, China, 2024: 388-392

附录1

表5 可执行语句模型示例

查询种类	可执行语句
时间段查询	query 〈relation〉 feed filter[present(. Trip **intersects** 〈time_interval〉)] consume;
范围查询	query 〈relation〉 feed filter [(**deftime**(. Trip **at** 〈location〉) intersects 〈time_interval〉)] consume;
最近邻居查询	let UnitsTrains = Trains feed **projectextendstream**[Id, Line, Up; UTrip: units(. Trip)]consume; let UTOrdered = UnitTrains feed extend[Mintime: minimum(deftime(. UTrip))] sortby[Mintime asc] consume; query UTOrdered_Rtree UTOrdered feed filter [(deftime(. UTrip) intersects 〈period〉)] knearest[UTrip, 〈object〉, 〈k〉]consume;
join查询	let 〈index_name〉 = 〈relation〉 **createrтree**[UTrip]; query 〈relation1〉 feed extend[Stretch: trajectory(. Trip **atperiods** 〈time_interval〉)] **loopjoin**[〈inedx_name〉 〈relation2〉 **windowintersects**[**bbox**(. Trip)] {second}] consume;
轨迹相似性查询	let 〈object1〉 = 〈relation〉 filter [(deftime(. Trip) intersects 〈time_interval1〉)] extract[Trip]; let 〈object1〉 = 〈relation2〉 filter [(deftime(. Trip) intersects 〈time_interval2〉)] extract[Trip]; **query dist_euclidean**(to_pointseq(〈object1〉), to_pointseq(〈object2〉)) ＜〈threshold〉;
异地经营查询	query 〈relation〉 feed filter[. Trip **present** 〈time_interval〉] extend[Stretch: trajectory(. Trip)] filter[not(. Stretch inside 〈operation_distinct〉 extract[GeoData]))] filter [not(. Stretch **intersects** 〈operation_distinct〉)] consume;
绕路查询	let 〈object1〉 = Edges **shortestpathlf** [Curve, (Stops feed filter[. Kind=‘start’] extract[Stop]), (Stops feed filter [. Kind=‘end’] extract[Stop])] extract[Curve]; let 〈object2〉 = [const pointseq value ((getstartpoin(object1)) (getendpoint(object1)))]; let 〈object3〉 = 〈relation〉 feed filter[. Trip present 〈time_interval〉] extract[Trip]; query **dist_origin_and_destination**(to_pointseq(object3), to_pointseq(object1)) ＜〈threshold〉;

表6 自然语言查询和可执行语句对示例

查询种类	自然语言查询	可执行语句
时间点查询	Which places did the trains go at 8am?	query Trains feed filter[. Trip present [const instant value “2020-11-20-8:00”]] extend[Pos: val(. Trip atinstant [const instant value “2020-11-20-8:00”])] project [Pos] consume;
时间段查询	Where were the taxis from 8am and 9am?	query Taxis feed filter[. Trip present [const periods value ((“2024-06-15-8:00” “2024-05-15-9:00” TRUE TRUE))]] extend[Stretch: trajectory(. Trip atperiods [const periods value ((“2024-06-15-8:00” “2024-06-15-9:00” TRUE TRUE))])] project[Stretch] consume;
范围查询	Which trains pass through the park “Tiergarten” between 8:00 and 10:00?	query Trains feed filter [(deftime(. Trip at tiergarten) intersects [const periods value ((“2020-11-20-8:00” “2020-11-20-10:00” TRUE TRUE))])] consume;
最近邻居查询	Find the 6 continuous nearest neighbors to train 100 between 10:00 and 13:00 o’clock.	let t100 = Trains feed filter[. Id=100] extract[Trip]; query UTOrdered_RTree UTOrdered feed filter [(deftime(. UTrip) intersects [const periods value ((“2020-11-20-10:00” “2020-11-20-13:00” TRUE TRUE))])] greeceknearest[UTrip, t100, 6] consume;
轨迹相似性查询	Did the train 5 and train 15 have the similar trajectory?	let t5 = Trains feed filter[. Id=5] extract[Trip]; let t15 = Trains feed filter[. Id=15] extract[Trip]; query dist_euclidean(to_pointseq(t5), to_pointseq(t15))<0. 5;
join查询	Which Taxis have the same trajectory?	query Taxis feed extend[Stretch: trajectory(. Trip)] {t1} Taxis feed extend[Stretch: trajectory(. Trip)] {t2} symmjoin [. Stretch_t1 intersects . . Stretch_t2] sort rdup consume;
异地经营查询	Please tell me if Taxi 3 has any cross-regional operations?	query Taxis feed filter[. Id=3] extend[Stretch: trajectory(. Trip)] filter[not(. Stretch inside (district feed filter [. Name=“Gaochun District”] extract[GeoData]))] filter [not(. Stretch intersects boundary((district feed filter [. Name = “Gaochun District”])))] consume;
绕路查询	Did the taxi 5 take detours?	let t5besttraj = Edges shortestpathlf [Curve, (Stops feed filter[. Id=5] filter[. Kind=‘start’] extract[Stop]), (Stops feed filter[. Id=5] filter[. Kind=‘end’] extract[Stop])] extract[Curve]; let t5besttrajseq = [const pointseq value ((getstartpoint(t5besttraj)) (getendpoint(t5besttraj)))]; let t5 = Taxis feed filter[. Id=5] extract[Trip]; query dist_origin_and_destination(to_pointseq(t5), to_pointseq(t5besttrajseq)) <0. 1;

WANG Xie-Yang, Ph. D. candidate. His main research interest is moving objects databases.

XU Jian-Qiu, Ph. D., professor. His main research interest is spatio-temporal data management.

LIU Meng-Yi, Ph. D. candidate. Her main research interest is spatial databases.

ZONG Chen-Chen, Ph. D. candidate. His main research interest is machine learning.

GAO Yun-Jun, Ph. D., professor. His main research interests include big data management and analysis, DB and AI integration.

Background

The explosive growth of moving objects data has led to an increasing number of users engaging in the usage and management of such data, which in turn has driven the rise of natural language interface to database (NLIDB). However, NLIDB primarily focuses on relational databases, and as SQL is the most commonly used query language, SQL lacks support for spatio-temporal operations. Existing systems rely on database optimizers for query optimization, posing challenges for mainstream large language model (LLM)-based approaches. On the other hand, translating to executable languages for spatio-temporal databases

can address these issues, but the Text-to-executable-language (Text-to-EXE) field is still in the early stages and lacks a unified evaluation standard.

High-quality benchmarks assess the performance of different NLIDB systems and promote continuous progress of NLIDB. Current benchmarks primarily focus on relational databases and lack real-world scenario validation in the moving objects database domain. Evaluation metrics emphasize translation accuracy, with less attention given to the efficiency of executable languages. Furthermore, LLMs have yet to undergo unified validation in the Text-to-EXE field regarding complex spatio-temporal queries and query optimization.

This paper proposes NALMOBench, a complex real-world benchmark for natural language query translation in moving object databases (MODs), which presents an LLM-based corpus generation method for MODs. NALMOBench includes common moving object queries as well as complex real-world scenario queries. The paper decomposes the Text-to-EXE problem into natural language understanding and structured language generation, and provides baseline methods. Experimental results show that NALMOBench presents significant challenges to existing Text-to-EXE methods and LLMs. These methods generally cannot directly control the optimization of the generated structured languages. The given baseline methods outperform existing methods and LLMs in both translation capability and query optimization.

This work was partially supported by the National Natural Science Foundation of China under Grant No. U23A20296 and 62472217, and Postgraduate Research & Practice Innovation Program of Jiangsu Province under Grant No. KYCX24_0608.