Computer Science and Application 计算机科学与应用, 2025, 15(8), 198-206
Published Online August 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.158210

Hans汉斯

基于大语言模型的地理信息查询系统

崔悦慧，高　悦

江苏省测绘工程院，江苏　南京

收稿日期：2025年7月14日；录用日期：2025年8月13日；发布日期：2025年8月21日

摘　要

针对传统地理信息系统(GIS)工具操作复杂、学习成本高的问题，本文提出一种融合规则引擎与大语言模型(LLM)的智能查询系统。通过动态路由算法实现查询任务的分流处理，结合领域知识增强的Prompt工程与PostGIS空间函数深度集成，解决了自然语言到空间SQL的语义转换难题。实验表明，系统使用江苏省生态产品价值实现工程研究中心平台数据，在空间连接查询中的准确率达72%，较通用模型提升25%以上，支持本地化部署保障地理敏感数据安全。

关键词

自然语言处理，地理信息系统，大语言模型，动态路由，PostGIS

Geographic Information Query System Based on Large Language Models

Yuehui Cui, Yue Gao

Jiangsu Province Surveying and Mapping Engineering Institute, Nanjing Jiangsu

Received: Jul. 14th, 2025; accepted: Aug. 13th, 2025; published: Aug. 21st, 2025

Abstract

To address the issues of complex operation and high learning costs associated with traditional Geographic Information System (GIS) tools, this paper proposes an intelligent query system that integrates a rule engine with a Large Language Model (LLM). The system achieves the shunting processing of query tasks through a dynamic routing algorithm, and combines domain knowledge-enhanced Prompt engineering with in-depth integration of PostGIS spatial functions, thereby solving the problem of semantic conversion from natural language to spatial SQL. The system utilizes platform data from the Jiangsu Engineering Research Center for Ecological Product Value Realization. Experiments show that the system achieves an accuracy rate of 72% in spatial join queries, which

文章引用: 崔悦慧, 高悦. 基于大语言模型的地理信息查询系统[J]. 计算机科学与应用, 2025, 15(8): 198-206.
DOI: 10.12677/csa.2025.158210

is more than 25% higher than that of general models, and supports localized deployment to ensure the security of geographically sensitive data.

Keywords

Natural Language Processing, Geographic Information System, Large Language Model, Dynamic Routing, PostGIS

1. 引言

地理信息数据规模呈指数级增长，其多源异构特性(遥感影像、矢量数据、社会经济统计表等)对分析工具提出更高要求。传统 GIS 依赖专业人员编写 SQL 或 Python 脚本，存在两大瓶颈：

1) 操作复杂性：非专业人员需数月培训才能掌握空间函数(如 ST_Intersects)的使用；

2) 语义鸿沟：自然语言中的“某区域地块面积统计”需手动转换为含 ST_Area 的 SQL 语句。

近年 NL2SQL 技术虽降低查询门槛，但在 GIS 领域存在显著局限：空间函数缺失：RAT-SQL [1]等通用模型忽略 PostGIS 函数；领域知识不足：同义词(“宗地”/“地块”)导致自然语言转换 SQL [2]字段映射失败；安全风险：云端 API 无法满足军事地理等敏感场景需求。

本文贡献：

1) 首创规则–模型动态路由架构，实现简单查询毫秒响应与复杂空间分析高精度转换；

2) 提出领域知识注入机制，通过字段注释与属性字典解决语义歧义；

3) 开发本地化部署方案，集成开源 AI 模型与 PostGIS，保障数据安全。

2. 地理信息领域创新及发展

2.1. NL2SQL 技术发展

自然语言转结构化查询语言(NL2SQL)技术旨在通过自然语言接口降低数据库查询门槛。早期研究主要依赖规则模板与关键词匹配方法。例如，Li 等[3]提出的 NaLIR 系统通过预定义语法规则实现简单查询的转换，但其句式灵活性差，难以处理复杂逻辑(如嵌套子查询)。随着深度学习的发展，统计学习方法逐渐成为主流。Zhong 等[4]提出的 Seq2SQL 模型，采用强化学习框架生成 SQL 语句，在 WikiSQL 数据集上实现 61%的准确率，但在多表关联场景中表现有限。

近年来，预训练语言模型(如 BERT、GPT [5])的兴起推动了 NL2SQL 技术的突破。Wang 等[1]提出的 RAT-SQL 模型通过关系感知的 Schema 编码技术，显著提升了复杂查询(含 JOIN 和聚合函数)的生成能力，在 Spider 数据集上达到 65%的执行准确率。然而，现有研究多聚焦于通用数据库场景，缺乏对空间数据库(如 PostGIS [6])的针对性优化。例如，在涉及地理空间函数(如 ST_Area、ST_Buffer)的查询中，通用模型常因领域知识缺失而生成错误 SQL。

2.2. 地理信息系统智能化

地理信息系统(GIS)的智能化转型是地理信息管理的核心趋势。传统 GIS 工具(如 ArcGIS、QGIS)依赖专业人员编写 Python 脚本或 SQL 语句，操作复杂且耗时。近年来，AI 与 GIS 的融合研究主要集中在

以下方向：

遥感影像分析：基于卷积神经网络(CNN)的自动地物分类[7]和变化检测技术，显著提升了影像解译效率。

空间数据挖掘：利用图神经网络(GNN)分析地理实体间的拓扑关系(如道路网络连通性)，辅助城市规划决策[8]。

自然语言交互：初步尝试将语音助手集成至 GIS 软件，但其功能局限于简单查询(如“显示地图图层”)，未支持复杂空间分析。

PostGIS 作为开源空间数据库扩展，提供了强大的空间计算能力(如 ST_Intersects、ST_Union)，但其使用仍需编写复杂 SQL 语句。现有研究尚未充分结合 NL2SQL 技术与 PostGIS 函数，导致非专业人员难以利用其空间分析潜力。例如，统计某区域地块面积时，用户需手动调用 ST_Area 函数，而通用 NL2SQL 工具无法自动识别此类需求。

本研究通过将大语言模型与 PostGIS 深度集成，首次实现了自然语言驱动的复杂空间查询，填补了 GIS 智能化在交互方式上的空白。

3. 系统设计与实现

3.1. 系统设计

本系统采用分层架构与混合式处理机制，构建了一套面向地理信息管理的智能查询与分析平台。其核心模块包括：1) AI 处理模块，集成本地 DeepSeek [9]模型与云端 KIMI API，通过领域知识增强的 Prompt 工程与动态路由机制，解决复杂查询(如多表 JOIN、空间函数嵌套)的语义理解与 SQL 生成问题；2) 数据库适配模块，封装 PostgreSQL/PostGIS 操作接口，支持跨数据库兼容与高并发访问，结合沙盒环境与参数化查询保障数据安全。

系统采用模块化分层设计，分为前端层、后端层和数据层，各层职责清晰且通过标准化接口通信，确保高内聚、低耦合。具体架构如图 1 所示：

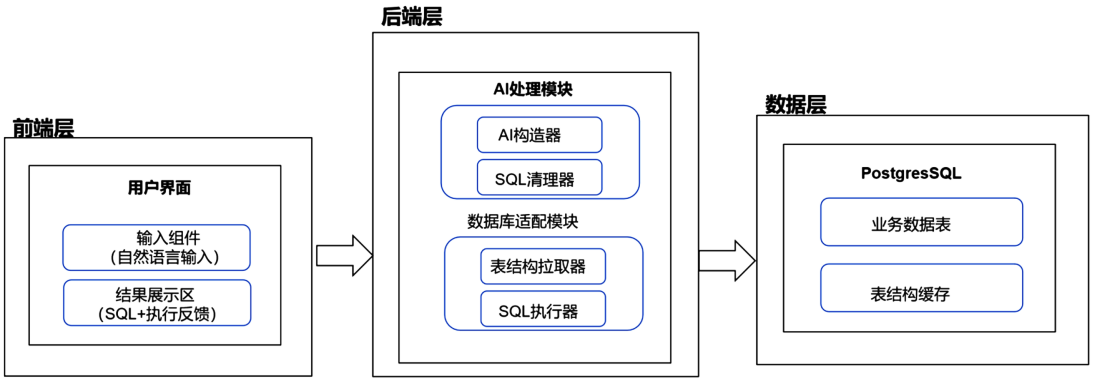

Figure 1. System architecture diagram
图 1. 系统架构图

3.1.1. 前端层

前端交互层基于 Vue3 框架构建，支持用户通过自然语言输入的方式与系统进行交互，支持多模态数据展示和会话管理功能。系统通过智能提示输入框实时推荐字段名与空间函数(如“缓冲区分析”)，支持自动补全(如字段名、函数名)与语法高亮，降低用户输入错误率；支持结果动态排序与筛选，通过历史记录模块自动缓存用户最近 30 天内的查询语句和结果，方便用户随时回顾和复用。为了确保数据传输的安

全性，系统采用了 HTTPS 加密通信技术，前端组件通过 REST API 与后端服务进行交互，确保低延迟与高响应性。如图 2 所示：

Figure 2. Frontend interface schematic
图 2. 前端界面示意图

3.1.2. 后端层

后端层分为三部分：人工智能(AI)处理模块、SQL 清理器模块、数据库适配模块。

人工智能处理模块是连接用户请求与多种人工智能在线 KIMI 模型、本地 deepseek 模型的核心组件，负责动态选择模型、处理输入/输出、优化生成结果，并集成消歧机制。该模块具体功能实现包括：

1) AI 构造器模块，该功能根据用户自然语言输入请求通过模型选择算法进行复杂度分析、模型策略配置和敏感数据检测，选择最优模型；复杂度分析，根据需求是否涉及多变连接或者空间分析进行判断，模型策略配置包括规则引擎模型，在线 AI 模型，本地 AI 模型。如不涉及空间分析且无多表连接等复杂查询时选择规则引擎方法进行数据库查询语句的生成，如涉及多表嵌套查询以及计算数据长度、面积等空间计算复杂查询选择调用 AI 模型，如涉及个人身份信息、地理信息分布信息、土地利用与规划数据等敏感场景数据选择本地模型处理。

其中动态路由算法采用复杂度评分公式，其中表示空间关键词数量(如“面积”“缓冲区”)，表示多表关联关键词数量(如“和”“对比”)，表示嵌套结构深度。路由选择伪代码如图 3 所示：

```
def route_query(query: str) -> str:
    # 关键词检测
    spatial_kws = ["面积", "相交", "缓冲区", "覆盖率"]
    join_kws = ["和", "与", "关联"]

    # 计算复杂度
    score = 0
    if any(kw in query for kw in spatial_kws):
        score += 2
    if any(kw in query for kw in join_kws):
        score += 1
    if "的" in query and ("统计" in query or "计算" in query):
        score += 1

    # 路由决策
    if score == 0:
        return rule_engine.generate(query)  # 正则模板匹配
    else:
        prompt = build_llm_prompt(query)    # 构造领域增强Prompt
        return llm.generate(prompt)         # 调用本地DeepSeek模型
```

Figure 3. Diagram of the route selection pseudocode
图 3. 路由选择伪代码图

2) 数据库查询语句得到后通过结果处理算法为数据库查询语句添加安全过滤、查询性能优化，使用禁用 DROP 策略，对 DROP 语句进行严格的过滤和控制，始终使用参数化查询避免 SQL 注入风险，严格控制普通用户权限；在 PostgreSQL 中，通过在查询中使用/* + Index (table_name index_name)*/语法来指定使用特定的索引，显式指定查询优化器使用特定索引的方式，确保查询以最优的方式执行。上下文管理，该功能为会话历史维护和缓存 Schema 注释，提升多轮对话连贯性。系统使用一个字典来存储每个用户的会话历史。通过会话 ID 作为字典的键，值是一个列表形式，用于存储该用户的查询历史和系统响应，在每次用户发送查询请求后，将用户的查询和系统生成的 SQL 语句、查询结果等信息添加到对应的会话历史中，列表设置最大长度限制为 10,000 字符，超出限制时删除最早的历史记录，在处理当前查询时，根据用户标识符从字典中检索该用户的会话历史，以便参考之前的对话内容，更好地理解和处理当前查询；在系统初始化时，使用数据库的系统表(如 information_schema)来查询从数据库中加载表结构和字段注释等 Schema 信息，并将其存储在内存中，这些信息数据库结构发生变化时，通过监听数据库结构变更事件来更新内存中的 Schema 注释缓存，在处理查询时，从内存中的缓存获取表和字段的注释信息，以帮助 AI 模型更好地理解和生成 SQL 查询，如图 4 所示。

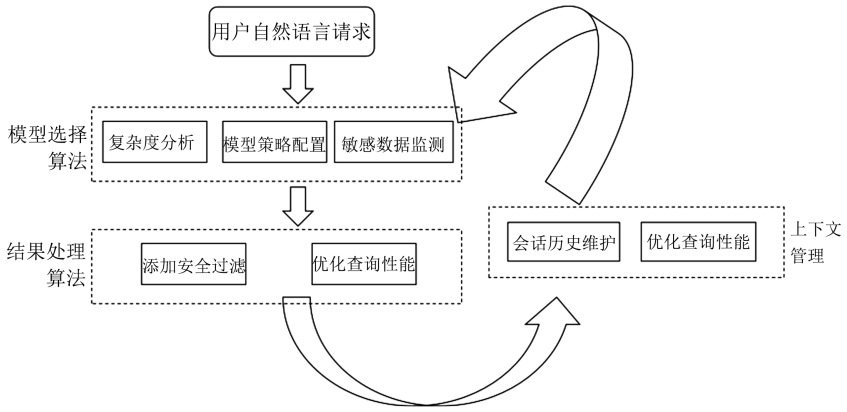

Figure 4. AI builder module schematic
图 4. AI 构造器模块示意图

3) SQL 清理器模块功能是负责确保 AI 生成的 SQL 安全、合法且高效，其核心功能包括安全过滤、语法验证、语义检查、性能优化和格式化输出。安全过滤(Security Filtering)系统定义关键字黑名单表，用于维护包含危险 SQL 关键字 DROP、DELETE、INSERT 等的列表，根据普通用户权限对生成的 SQL 进行扫描，发现包含黑名单关键字的语句直接拦截；操作白名单，定义普通用户允许的 SELECT、JOIN、WHERE 操作类型，从源头上杜绝危险操作；参数化查询，对用户输入值进行单引号转义的方法阻止危险操作(如 DROP、DELETE)和 SQL 注入。语法验证，系统通过利用 PostgreSQL 的 EXPLAIN 命令，在安全的沙盒环境中对 SQL 语句进行预执行，检查语法是否正确，同时避免实际修改数据库数据；使用 sqlparse 库，解析 SQL 结构树，确保 SQL 符合 PostgreSQL 语法规范。语义检查，通过访问数据库的 information_schema 系统视图，获取数据库的元数据信息，验证 SQL 语句中的表名、字段名是否存在，将查询到的 Schema 信息进行缓存，减少对数据库元数据的实时查询次数，提高处理效率，减少实时查询开销；性能优化，根据查询条件和数据库统计信息，添加索引提示，引导数据库优化器选择最优的索引；并对 SQL 语句进行重写，将复杂子查询转换为连接查询提升执行效率。格式化输出，通过 AI 处理模块确保使用 AI 大语言模型自动生成的数据库查询语句符合 PostgresSQL 格式要求，按照统一的格式规范对 SQL 语句的缩进、换行、关键字大写等进行格式化，使 SQL 语句更易读。模块示意如图 5 所示：

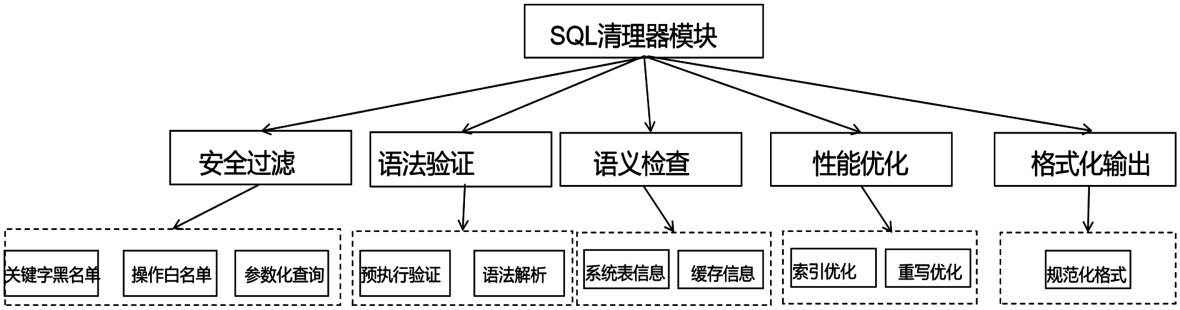

Figure 5. SQL cleaner module schematic
图 5. SQL 清理器模块示意图

数据库适配模块是系统与数据库交互的核心组件，包含表结构拉取器和SQL执行器，支持PostgreSQL数据库。

表结构拉取器通过查询数据库的系统表 information_schema 和 pg_catalog，动态获取数据库中的表、字段、主键、索引等元数据。在内部根据具体的数据库类型执行相应的查询语句，但对外提供的方法签名和返回数据格式保持一致，从而实现数据库操作的统一接口。

4) SQL 执行器负责执行具体的 SQL 语句，并将执行结果返回。通过与数据库建立连接，发送 SQL 语句，并接收和处理返回的结果集。执行器处理各种类型的SQL语句，包括查询(SELECT)、插入(INSERT)、更新(UPDATE)和删除(DELETE)等操作。模块如图 6 所示：

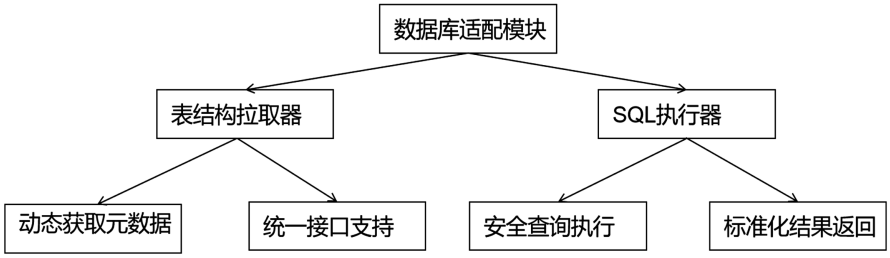

Figure 6. Database adaptation module schematic
图 6. 数据库适配模块示意图

3.1.3. 数据层

地理信息矢量数据的业务数据表，通过程序进行批量存储，表结构字段注释管理格式化录入。表结构与注释缓存机制为提升系统性能与响应速度，数据层采用动态表结构缓存技术，实现对 PostgreSQL 数据库元数据的高效管理。

初始化加载：系统启动时，通过查询 information_schema 和 pg_catalog 系统表，全量拉取表结构并构建内存缓存。结合 geometry_columns 系统表识别空间数据表，记录几何类型与坐标系参数。减少数据库压力：避免每次生成 SQL 时重复查询元数据。加速语义解析：规则引擎与 AI 模型可直接从缓存获取字段注释，辅助自然语言消歧。

矢量数据入库，使用 shp2pgsql 命令行工具，将 Shapefile 转换为 PostgreSQL 支持的 SQL 脚本。

字段注释管理是连接自然语言与数据库语义的核心桥梁，系统通过以下方式实现高效管理

1) 注释定义与存储，数据库原生支持：通过 PostgreSQL 的 COMMENT 语法为字段添加描述：COMMENT ON COLUMN land_parcels.area IS'地块面积(平方米)'；集中化存储：注释信息存储在 pg_description 系统表中，便于统一查询。

2) 注释自动化注入动态加载：表结构缓存加载时，同步拉取字段注释，构建「字段名－注释」映射字典。AI 提示增强：生成 SQL 时，将注释作为上下文注入大模型 Prompt

3) 注释批量维护，YAML 配置文件：将注释定义在 YAML 文件中，通过脚本批量执行更新，如图 7 所示。

```yaml
#数据库表注释配置文件
land parcels:
    table comment:"土地宗地信息表"
    columns:
        geom:"几何边界(WGS84坐标系)"
        area:"地块面积(平方米)"
        parcel id:"宗地唯一标识符"
        owner name:"土地所有者姓名"
        registration date:"登记日期"
        land use type:"土地用途类型"
forest coverage:
    table comment:"森林覆盖率数据表"
    columns:
        region name:"地区名称"
        year:"数据年份"
        forest area:"森林面积(平方米)"
        geom:" 地区多边形几何数据"
        data source:"数据来源说明"
        update time:"数据更新时间"
arable land:
    table comment:"耕地面积数据表"
    columns:
        region name:"地区名称"
        year:"数据年份"
        arable area:"耕地面积(平方米)"
        soil quality:"土壤质量等级"
        geom:"耕地多边形几何数据"
        data source:"数据来源说明"
        update time:"数据更新时间"
```

Figure 7. Database table annotation configuration schematic
图 7. 数据库表注释配置文件示意图

自动化同步：执行 python update_comments.py 后，自动遍历 YAML 文件并更新数据库注释。

坐标系统一：入库后调用 ST_Transform (geom, 4326)强制转换坐标系，确保分析一致性。

3.2. 系统交互流程

本章介绍系统交互具体过程，如下所述：

1) 用户输入：用户在前端界面输入自然语言请求，例如“显示 2023 年北京的森林覆盖率”或“分析某区域的土地利用类型变化趋势”。前端将这些请求发送至 FastAPI 的接口。

2) 路由决策：后端接收到请求后，首先对查询的复杂度进行评估。对于简单的查询(如直接查询单一表的数据)，系统使用规则引擎快速生成 SQL 语句。对于复杂的查询(如涉及多表关联、空间分析或数据聚合)，系统则调用 AI 模型生成 SQL 语句。AI 模型会根据输入的自然语言请求，结合数据库结构和历史查询模式，生成最优的 SQL 查询语句。

3) 空间计算：如果查询涉及空间数据操作(如计算缓冲区、叠加分析、几何关系判断等)，系统会调用空间分析接口，该接口封装了 PostGIS 函数，能够执行各种空间分析任务。系统将空间计算的结果作为几何操作结果返回给前端。

4) 前端接收到后端返回的 JSON 数据后，进行解析将生成的数据库查询语句和查询结果进行显示，如果返回查询结果未达到用户预期，用户可以根据 SQL 语句和返回结果修正提问方式后再次提问。

为确保系统的安全性，全链路采用 HTTPS 加密通信，所有数据传输都在加密通道中进行，有效防止数据泄露和中间人攻击。敏感数据(如数据库凭据、API 密钥等)通过 HashiCorp Vault 进行动态管理，实现了集中化的安全管理。系统在设计时充分考虑了扩展性，通过抽象数据库接口层，使得系统能够轻松适配 MySQL、Oracle 等其他支持空间扩展的数据库，降低了系统的耦合度，提高了系统的可维护性和可扩展性。

4. 实验评估

4.1. 实验数据

本系统测试数据来自江苏省生态产品价值实现工程研究中心平台，该平台针对江苏省自然资源状况，采集了包括林地、湿地、草地、城镇等陆域主要生态系统实物量，数据具有科学性、规范性，实验中提取该平台 500 条自然语言作为测试集评估模型的正确率。

4.2. 空间查询准确性对比

针对是否使用动态路由机制进行实验，实验结果如表 1 所示。

Table 1. Comparison of spatial query accuracy
表 1. 空间查询准确性对比

查询类型	本系统正确率	无动态路由正确率	提升率
单表空间过滤	98%	95%	3%
多表空间连接	75%	62%	23%
嵌套面积计算	72%	47%	25%

5. 结论与展望

本系统通过规则－模型混合架构，解决了纯规则方法灵活性不足与纯模型方法效率低下的矛盾。此外，字段注释与属性字典的引入，将领域知识显式注入 LLM，弥补了通用模型在专业场景中的语义鸿沟。本地大模型部署需较高 GPU 资源(峰值内存 12 GB)；多轮对话支持有限，需进一步集成上下文管理模块。

本研究提出了一种基于大语言模型的地理信息查询系统，通过 Text2SQL 与 PostGIS 的深度集成，实现了非专业人员对地理数据的智能分析。未来工作将聚焦于：引入 Few-shot Learning 动态扩展属性字典；优化模型量化压缩技术，降低资源消耗；支持多轮对话上下文理解，增强交互连贯性。

参考文献

[1] Wang, B., Shin, R., Liu, X., Polozov, O. and Richardson, M. (2020) RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-To-SQL Parsers. *Proceedings of the* 58*th Annual Meeting of the Association for Computational Linguistics*, July 2020, 7567-7578. https://doi.org/10.18653/v1/2020.acl-main.677

[2] Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J., Liu, T., *et al.* (2019) Towards Complex Text-To-SQL in Cross-Domain Database with Intermediate Representation. *Proceedings of the* 57*th Annual Meeting of the Association for Computational Linguistics*, July 2019, 4524-4535. https://doi.org/10.18653/v1/p19-1444

[3] Li, F. and Jagadish, H.V. (2016) NaLIR: An Interactive Natural Language Interface for Querying Relational Databases. *Proceedings of the* 2014 *ACM SIGMOD International Conference on Management of Data*, Snowbird, 22-27 June 2014, 709-721.

[4] Zhong, V., Xiong, C.M. and Socher, R. (2017) Seq2SQL: Generating Structured Queries from Natural Language Using Reinforcement Learning. arXiv: 1709.00103.

[5] Achiam, J., *et al.* (2023) GPT-4 Technical Report. arXiv: 2303.08774.

[6] Strobl, C. (2016) PostGIS. In: Shekhar, S., Xiong, H. and Zhou, X., Eds., *Encyclopedia of GIS*, Springer, 1-8. https://doi.org/10.1007/978-3-319-23519-6_1012-2

[7] Zhang, C., Zhang, L.F. and Du, B. (2021) Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art. *IEEE Geoscience and Remote Sensing Magazine*, **4**, 22-40.

[8] Liu, Y., *et al.* (2022) Graph Neural Networks for Spatial Networks: A Survey. *ACM Computing Surveys*, **55**, 1-36.

[9] Piplani, T. and Bamman, D. (2018) DeepSeek: Content Based Image Search & Retrieval. arXiv: 1801.03406.