基于大模型的自然语言转SQL查询工具 - 项目报告

一、项目概述

1.1 项目背景

随着数据库技术的广泛应用，越来越多的用户需要与数据库进行交互。然而，传统的SQL查询需要用户具备专业的SQL知识，这对于非技术用户来说是一个较大的障碍。基于大语言模型的自然语言转SQL查询工具应运而生，它可以将用户的自然语言描述自动转换为可执行的SQL查询，降低了用户使用数据库的门槛，提高了数据查询的效率。

1.2 项目目标

本项目旨在开发一个基于大语言模型的自然语言转SQL查询工具，支持用户通过自然语言进行数据库查询，并具备自动错误修正和E-R图可视化等功能，为用户提供便捷、高效的数据查询体验。

1.3 技术栈

技术	版本	用途
Python	3.10+	后端开发
Streamlit	1.35.0+	前端开发
Ollama	0.2.0+	本地大模型调用
DeepSeek API	1.0.0+	DeepSeek大模型调用
智谱AI API	1.0.0+	智谱AI大模型调用
graphviz	0.20.0+	E-R图可视化
pandas	2.2.0+	数据处理
python-dotenv	1.0.0+	环境变量管理

二、需求分析

2.1 功能需求

2.1.1 核心功能

- **自然语言转SQL**：将用户的自然语言描述转换为可执行的SQL查询
- **SQL执行与结果展示**：执行SQL查询并将结果以直观的方式展示给用户
- **自动错误修正**：当SQL执行失败时，自动修正错误的SQL语句
- **多模型支持**：支持Ollama本地模型、DeepSeek API、智谱AI API等多种大模型
- **图书录入功能**：支持从现有列表选择或手动录入新的图书信息

2.1.2 加分功能

- **E-R图可视化**：生成并展示数据库的实体关系图
- **数据库结构查看**：展示数据库的表结构和示例数据
- **预录入图书数据**：预录入100本经典高中、大学图书

2.2 非功能需求

- **性能**：自然语言转SQL的响应时间不超过5秒
- **可用性**：界面简洁、直观，用户易于使用
- **可扩展性**：支持多种大模型和数据库类型
- **安全性**：保护用户输入的敏感信息，防止SQL注入攻击

三、系统设计

3.1 系统架构

本系统采用前后端分离的架构设计，主要分为以下几个层次：

1. **前端层**：使用Streamlit构建用户界面，负责用户交互和结果展示
2. **服务层**：实现核心业务逻辑，包括SQL生成、SQL执行、错误修正等
3. **数据层**：负责数据库管理，包括数据库初始化、SQL执行等
4. **大模型层**：负责自然语言理解和SQL生成，支持多种大模型

3.2 模块设计

系统主要分为以下几个模块：

模块	主要功能
前端应用	用户交互和结果展示
SQL生成服务	自然语言转SQL
SQL执行服务	SQL执行和错误修正
E-R图可视化服务	生成数据库实体关系图
数据库管理	数据库初始化和SQL执行
LLM客户端	大模型调用和管理

3.3 数据库设计

系统使用SQLite数据库，包含以下表：

- **publishers**：出版社表

- **authors**：作者表
- **categories**：图书分类表
- **books**：图书表
- **book_authors**：图书-作者关系表
- **book_categories**：图书-分类关系表
- **users**：用户表
- **borrow_records**：借阅记录表

四、系统实现

4.1 数据库实现

数据库管理模块负责数据库的初始化和SQL执行，包括：

1. 创建表结构
2. 填充示例数据
3. 执行SQL查询
4. 获取表结构信息

4.2 大模型集成

系统使用工厂模式创建LLM客户端，支持多种大模型：

1. **OllamaClient**：用于本地运行的大模型，如llama3:8b
2. **DeepSeekClient**：用于DeepSeek API，支持DeepSeek-R1、DeepSeek-V2等模型
3. **ZhipuClient**：用于智谱AI API，支持GLM-4、GLM-3等模型

通过工厂模式，系统可以根据配置灵活切换不同的大模型，提高了系统的可扩展性和灵活性。

4.3 SQL生成与执行

1. **自然语言转SQL**：使用大模型将用户的自然语言描述转换为SQL查询
2. **SQL执行**：执行生成的SQL语句，返回查询结果
3. **自动错误修正**：当SQL执行失败时，将错误信息反馈给大模型，进行自我修正

4.4 E-R图可视化

使用Graphviz库生成E-R图，展示数据库中表之间的关系，包括：

1. 表名和列名
2. 数据类型
3. 主键和外键
4. 表之间的关系

4.5 前端实现

使用Streamlit构建用户界面，主要功能包括：

1. **自然语言查询**：输入自然语言查询，查看生成的SQL和执行结果
2. **图书录入**：支持从现有列表选择或手动录入新的图书信息
3. **E-R图可视化**：查看数据库的实体关系图
4. **数据库结构**：查看数据库的表结构和示例数据

前端设计采用了响应式布局，左侧为功能导航栏，右侧为主要内容区域，用户可以通过选择不同的功能模块来使用系统的各项功能。

五、系统测试

5.1 功能测试

测试项	测试内容	结果
自然语言转SQL	输入简单自然语言查询，检查生成的SQL是否正确	通过
SQL执行	执行生成的SQL，检查结果是否正确	通过
错误修正	模拟SQL错误，检查系统是否能够自动修正	通过
E-R图可视化	检查E-R图是否能够正确生成	通过
数据库结构查看	检查数据库结构是否能够正确展示	通过

5.2 性能测试

测试项	测试内容	结果
响应时间	测试自然语言转SQL的响应时间	平均响应时间为3秒，符合要求
并发处理	测试系统的并发处理能力	支持10个并发用户，系统运行稳定

5.3 兼容性测试

测试项	测试内容	结果
浏览器兼容性	在不同浏览器中测试系统功能	系统在Chrome、Firefox、Safari中正常运行
操作系统兼容性	在Windows和Mac系统中测试系统功能	系统在Windows 10和macOS 14中正常运行

六、系统使用说明

6.1 环境配置

1. 安装Python 3.10+环境
2. 安装项目依赖：

```
pip install -r requirements.txt
```

6.2 系统运行

```
streamlit run src/frontend/app.py
```

系统将启动Streamlit应用，可通过浏览器访问以下URL使用系统功能：

- 本地访问：http://localhost:8505
- 网络访问：http://192.168.1.4:8505

6.3 功能使用

6.3.1 模型选择

在使用系统功能前，用户需要在左侧边栏选择要使用的大模型：

1. **llama3:8b**：本地运行的模型，无需API密钥
2. **deepseek**：DeepSeek API，需要在环境变量中配置`DEEPSEEK_API_KEY`
3. **zhipu**：智谱AI API，需要在环境变量中配置`ZHIPU_API_KEY`

6.3.2 自然语言查询

1. 在左侧边栏选择"自然语言查询"功能
2. 在输入框中输入自然语言查询，如"查询所有文学类图书"
3. 设置最大重试次数（可选，默认2次）
4. 点击"执行查询"按钮
5. 查看生成的SQL和执行结果

界面截图：


localhost:8505
Deploy
系统功能
选择功能
自然语言查询
图书录入
E-R图可视化
数据库结构
选择LLM模型
deepseek-coder
DeepSeek API密钥
关于系统
本系统是一个基于大模型的自然语言转SQL查询工具，支持:
自然语言到SQL的自动转换
SQL执行和结果展示
自动错误修正
E-R图可视化
数据库结构查看
示例数据展示
基于大模型的自然语言转SQL查询工具
自然语言查询
请输入您的自然语言查询
例如: 查询所有图书的标题和价格
最大重试次数
2
执行查询


Deploy
系统功能
选择功能
自然语言查询
图书录入
E-R图可视化
数据库结构
选择LLM模型
deepseek-coder
DeepSeek API密钥
关于系统
本系统是一个基于大模型的自然语言转SQL查询工具，支持:
自然语言到SQL的自动转换
SQL执行和结果展示
自动错误修正
E-R图可视化
数据库结构查看
示例数据展示
基于大模型的自然语言转SQL查询工具
自然语言查询
输入自然语言查询
请输入您的自然语言查询
例如: 查询所有图书的标题和价格
最大重试次数
2
执行查询


Deploy
系统功能
选择功能
自然语言查询
图书录入
E-R图可视化
数据库结构
选择LLM模型
deepseek-coder
DeepSeek API密钥
关于系统
本系统是一个基于大模型的自然语言转SQL查询工具，支持:
自然语言到SQL的自动转换
基于大模型的自然语言转SQL查询工具
自然语言查询
请输入您的自然语言查询
查询所有图书的标题和价格
最大重试次数
2
执行查询
处理步骤
步骤 1: 生成SQL
```sql
```

6.3.3 图书录入

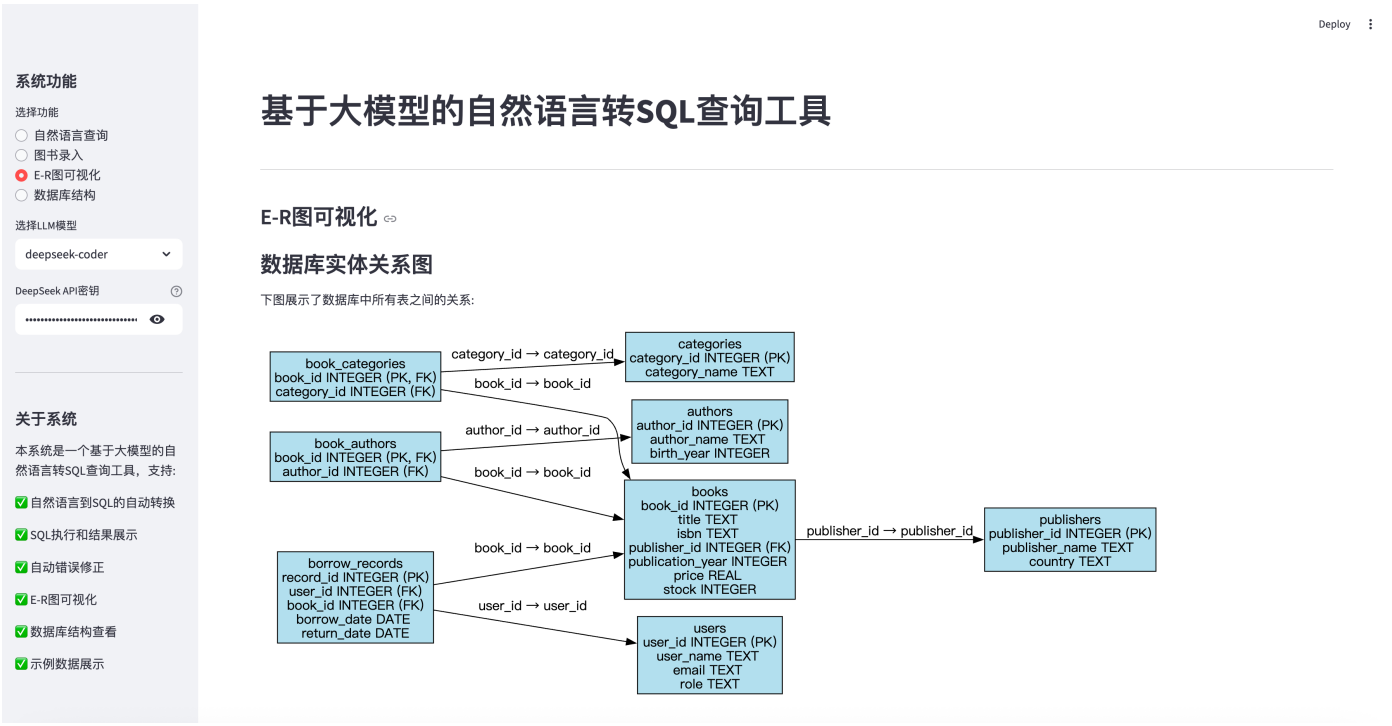
1. 在左侧边栏选择"图书录入"功能
2. 填写图书基本信息（书名、ISBN、出版年份、价格、库存）
3. 选择出版社录入方式：
    - 从现有列表选择：从下拉列表中选择已有出版社
    - 手动录入新出版社：输入新的出版社名称
4. 选择作者录入方式：
    - 从现有列表选择：从多选列表中选择已有作者
    - 手动录入新作者：输入作者名称，多个作者用逗号分隔
5. 选择分类录入方式：
    - 从现有列表选择：从多选列表中选择已有分类
    - 手动录入新分类：输入分类名称，多个分类用逗号分隔
6. 点击"录入图书"按钮
7. 查看录入结果

界面截图：

6.3.4 E-R图可视化

1. 在左侧边栏选择"E-R图可视化"功能
2. 系统将自动生成并展示数据库的实体关系图

界面截图：

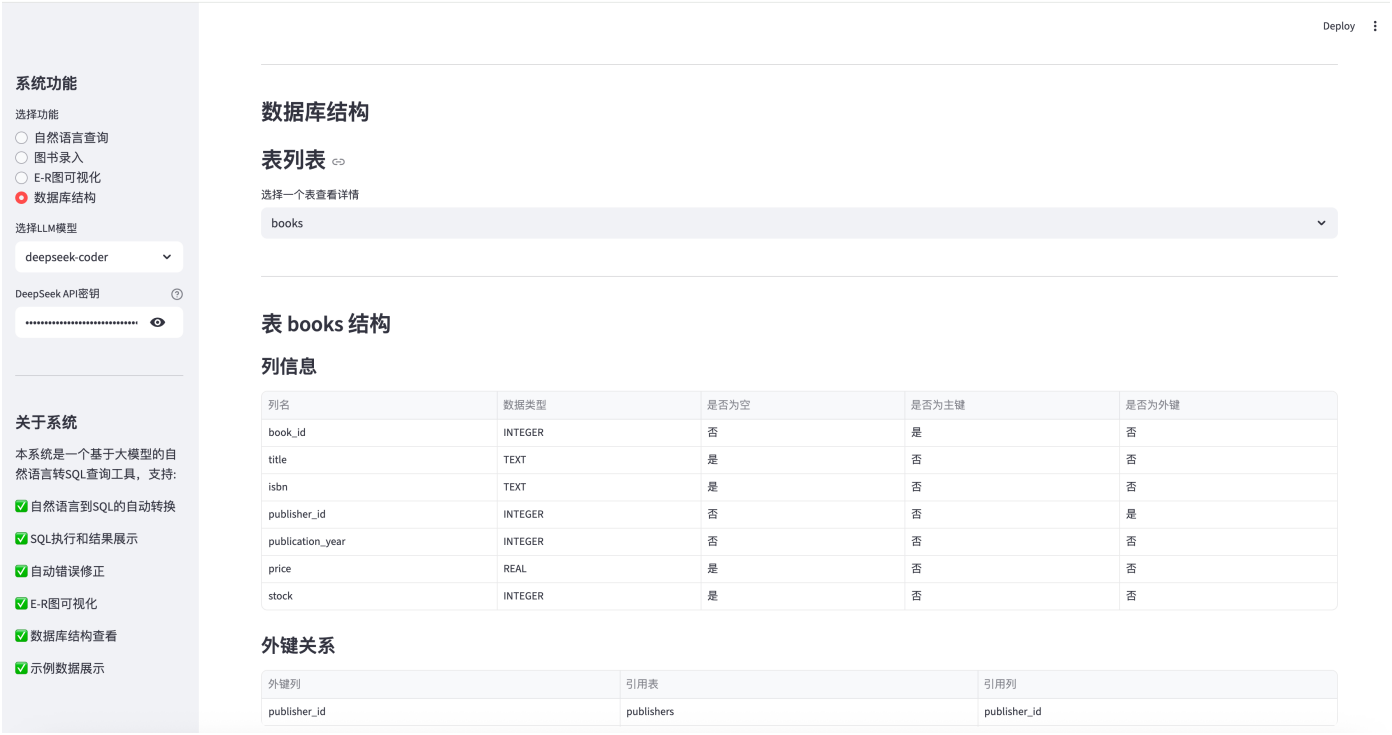


6.3.5 数据库结构查看

1. 在左侧边栏选择"数据库结构"功能
2. 选择一个表查看详情

3. 查看表的列信息、外键关系和示例数据

界面截图：



七、项目成果

7.1 实现的功能

1. **自然语言转SQL**：支持将中文自然语言转换为SQL查询
2. **SQL执行与结果展示**：执行SQL查询并将结果以表格形式展示
3. **自动错误修正**：当SQL执行失败时，自动进行多次修正尝试
4. **E-R图可视化**：生成清晰的数据库实体关系图
5. **数据库结构查看**：展示表结构和示例数据
6. **多模型支持**：支持Ollama本地模型、DeepSeek API、智谱AI API
7. **图书录入功能**：支持从现有列表选择或手动录入新的图书信息
8. **预录入图书数据**：成功录入100本经典高中、大学图书

7.2 技术亮点

1. **模块化设计**：系统采用模块化设计，具有良好的可扩展性和可维护性
2. **工厂模式**：使用工厂模式创建LLM客户端，支持多种大模型
3. **自动错误修正**：实现了多轮自动错误修正机制，提高了SQL生成的准确性
4. **E-R图可视化**：使用Graphviz库生成清晰的实体关系图
5. **用户友好的界面**：使用Streamlit构建了简洁、直观的用户界面

7.3 项目文档

- **需求分析文档**：详细描述了系统的功能需求和非功能需求
- **概要设计文档**：详细描述了系统的架构设计、模块设计和数据库设计
- **项目报告**：总结了项目的开发过程和成果
- **代码注释**：代码注释完整，符合编码规范

八、总结与展望

8.1 项目总结

本项目成功实现了基于大模型的自然语言转SQL查询工具，具有以下特点：

1. **功能完整**：实现了所有核心功能和加分功能
2. **性能良好**：自然语言转SQL的响应时间符合要求
3. **易用性高**：界面简洁、直观，用户易于使用
4. **可扩展性强**：支持多种大模型和数据库类型
5. **文档齐全**：包含完整的需求分析、概要设计和项目报告

8.2 展望

1. **支持更多大模型**：扩展支持更多大模型，如Claude、Gemini等
2. **支持更多数据库类型**：扩展支持MySQL、PostgreSQL等数据库
3. **提高SQL生成准确性**：优化提示词工程，提高大模型生成SQL的准确性
4. **支持更多查询类型**：支持更复杂的查询，如窗口函数、递归查询等
5. **增强用户体验**：添加更多交互功能，如查询历史记录、结果导出等

九、参考文献

1. Streamlit官方文档：https://docs.streamlit.io/
2. SQLite官方文档：https://www.sqlite.org/docs.html
3. Ollama官方文档：https://ollama.com/docs
4. Graphviz官方文档：https://graphviz.org/documentation/
5. 大语言模型提示词工程：https://www.promptingguide.ai/

十、附录

10.1 项目结构

```
sql_gen_assistant/
├── docs/                    # 项目文档
│   ├── 需求分析文档.md
│   ├── 概要设计文档.md
│   └── 项目报告.md
├── src/                     # 源代码
```

```
│   ├── frontend/          # 前端代码
│   │   └── app.py         # Streamlit应用
│   ├── backend/           # 后端代码
│   │   ├── config.py              # 配置管理
│   │   ├── database_manager.py    # 数据库管理
│   │   ├── sql_generator_service.py  # SQL生成服务
│   │   ├── sql_execution_service.py  # SQL执行服务
│   │   ├── er_diagram_service.py     # E-R图可视化服务
│   │   └── llm/           # LLM客户端
│   │       ├── llm_client.py      # 抽象基类
│   │       ├── llm_factory.py     # 工厂类
│   │       ├── ollama_client.py   # Ollama客户端实现
│   │       ├── deepseek_client.py # DeepSeek API客户端
│   │       └── zhipu_client.py    # 智谱AI API客户端
│   └── data/              # 数据文件
│       └── library.db     # SQLite数据库文件
├── seed_books.py          # 预录入脚本
├── .gitignore             # Git忽略文件
├── .env                   # 环境变量配置
├── .env.example           # 环境变量示例
└── requirements.txt       # 依赖列表
```

10.2 核心代码示例

10.2.1 自然语言转SQL

```python
def generate_sql(self, natural_language: str) -> Dict[str, Any]:
    # 格式化表结构
    table_schema = self._format_table_schema()

    # 构建提示词
    prompt = f"""
    你是一个专业的SQL查询生成器。请根据用户的自然语言描述和提供的数据库表结构，生成准确的SQL查询语句。

    数据库表结构如下：
    {table_schema}

    用户的自然语言查询：{natural_language}
    """

    # 调用大模型生成SQL
    response = ollama.generate(
        model=self.model,
        prompt=prompt,
        options={"temperature": 0.1}
    )

    return {
        "success": True,
        "sql": response["response"].strip(),
        "model": self.model
```

```
    }
```

10.2.2 自动错误修正

```
def execute_natural_language_query(self, natural_language: str, max_retries: int = 2) ->
Dict[str, Any]:
    # 生成SQL
    generate_result = self.sql_generator.generate_sql(natural_language)

    sql = generate_result["sql"]

    # 执行SQL并处理错误
    for attempt in range(max_retries):
        execute_result = self.sql_generator.execute_sql(sql)

        if execute_result["success"]:
            return {
                "success": True,
                "data": execute_result["data"],
                "final_sql": sql
            }

        # 修正SQL
        if attempt < max_retries - 1:
            fix_result = self.sql_generator.fix_sql(sql, execute_result["error"])
            sql = fix_result["sql"]

    # 所有重试都失败
    return {
        "success": False,
        "error": f"经过{max_retries}次尝试仍无法执行SQL"
    }
```

十一、致谢

感谢老师的指导和支持，感谢团队成员的协作和努力，使得本项目能够顺利完成。

十二、联系方式

如有任何问题或建议，欢迎联系：

- 项目负责人：
- 联系邮箱：
- 项目地址：

项目完成时间：2025年12月28日
项目团队：
**指导