1 引言

2 数据集和知识图谱构建

2.1 数据集介绍

在本项目中, 我们选择了 TMDB 作为我们的数据源. 现有的主要的电影数据源包括 TMDB, OMDb, IMDb, MovieLens 等, 其中 TMDB 在电影数据 API 中以其丰富的数据、开放的社区驱动模式和高质量的 API 而著称. 它不仅适用于电影和电视节目的信息查询和展示, 还提供了强大的推荐和搜索功能, 是开发电影相关应用的理想选择. 相比其他 API, 如 IMDb 和 OMDb, TMDB 在数据的实时更新、社区参与和免费使用方面具有明显的优势.

数据集的爬取主要分为搜索和获取详细信息两步. 首先, 通过在 `"https://api.themoviedb.org/3/movie/popular?api_key={api_key}&page={page}"` 指定 `api_key` 和 `page` (取值于 1 - 100), 我们可以获得 2000 部电影的简要信息; 然后, 在 `"https://api.themoviedb.org/3/movie/{movie_id}?api_key={api_key}&append_to_response=credits"` 中指定 `api_key` 和 `movie_id` (来源于上一步的简要信息), 可以获得这些电影的详细信息.

获取数据之后, 我们进行了一系列探索性分析. 首先, 我们对一些数据项统计了频数分布, 见 Figure 1. 在 `belongs_to_collection` 中, 排名靠前的是《星球大战》、《碟中谍》、《詹姆斯·邦德》、《哈利·波特》等经典系列电影; 在 `genres` 中, 排名靠前的是动作、剧情、冒险、恐怖、戏剧等电影类型; 在 `production_companies` 中, 排名靠前的是华纳兄弟、环球影业、哥伦比亚影业等电影制片公司; 而在 `original_country` 中, 美国一骑绝尘, 有超过 1400 部电影的原产地是美国.

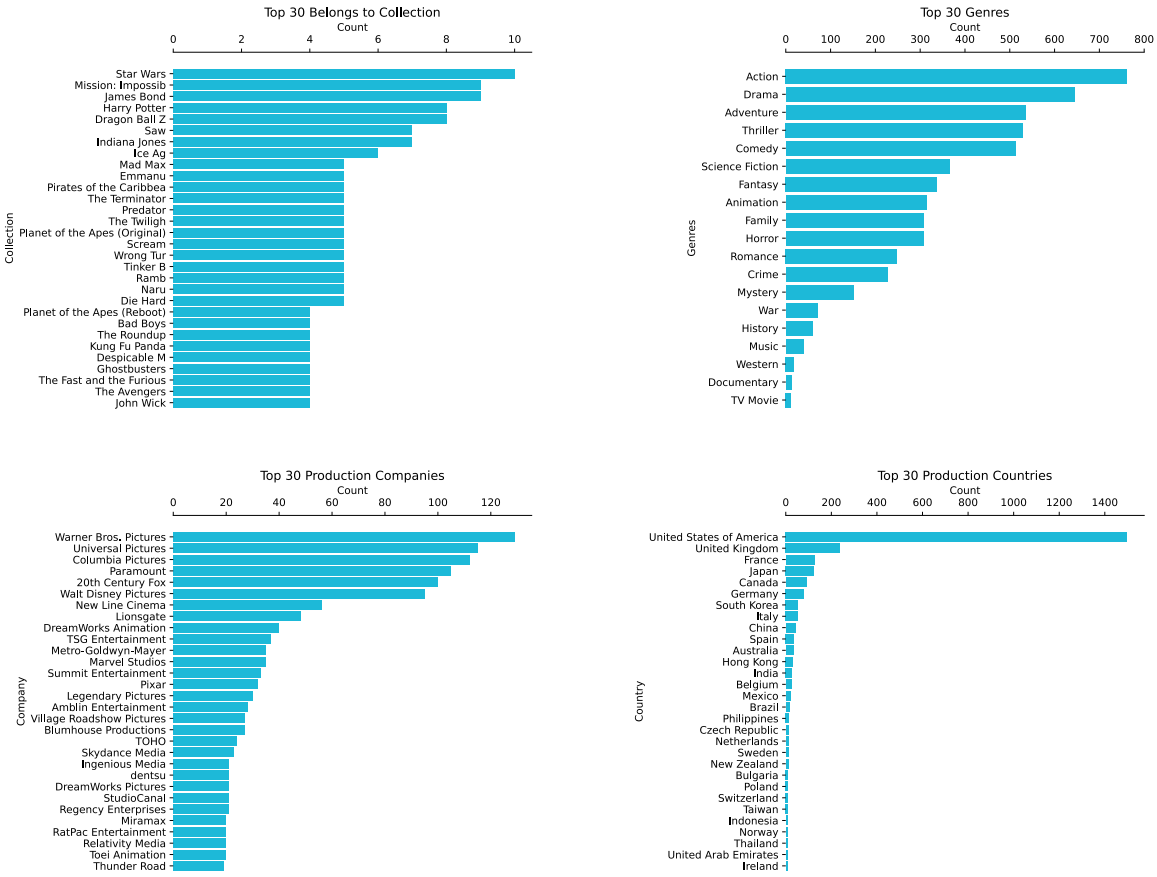

Figure 1: 部分数据项的频数分布

另外, 我们也通过散点图绘制了一些数值型数据之间的关系, 见 Figure 2. 在预算-收入散点图中, 发现预算存在明显的取整和趋同倾向, 大多数电影能够盈利. 在评分数-平均评分散点图中, 观察到评分数越多, 平均评分越高, 说明这些电影本身较受欢迎, 能够得到较多较高的评分.

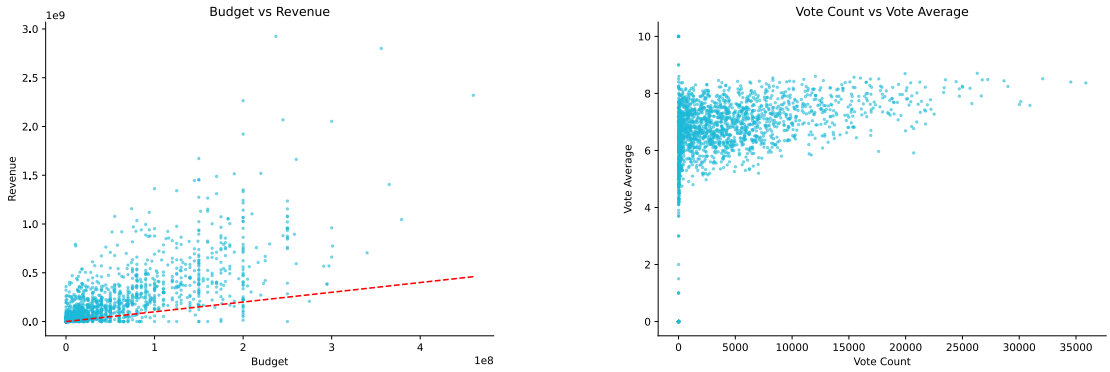

Figure 2: 部分数据项的频数分布

2.2 知识图谱构建

由于从 API 获取的数据已经是结构化的 json 数据, 我们的主要工作在于对其中键值的筛选和处理, 以使其符合数据库的存储要求. 具体地, 我们选择了一些关键字段, 例如电影的所属系列、预算、ID、概述、上映日期、收入、时长、标语和标题等构成了电影的基本信息节点; 从制作公司列表中提取公司的 ID、名称和所属国家, 构成制作公司节点; 提取电影的类型信息, 并将每个类型的名称存储到类型节点中. 在工作人员信息的处理上, 我们仅保留导演、制片人和执行制片人这几类重要职位, 并从工作人员列表中提取他们的 ID、姓名和职位, 构成工作人员节点. 对于演员信息, 我们选择前十名主要演员, 并提取他们的 ID、姓名、角色和性别信息, 构成演员节点. 处理前后的数据示例见 ./assets/raw_data_eg.md 和 ./assets/processed_data_eg.md.

然后, 我们将处理后的数据存入 Neo4j 数据库. 通过 `Neo4jGraph.get_schema` 方法, 输出数据库的架构如下:

```
Node properties:
Movie {budget: INTEGER, id: INTEGER, title: STRING, tagline: STRING,
belongs_to_collection: STRING, runtime: INTEGER, overview: STRING, revenue: INTEGER,
release_date: STRING}
Genre {name: STRING}
ProductionCompany {origin_country: STRING, name: STRING, id: INTEGER}
Country {name: STRING}
Person {name: STRING, id: INTEGER, gender: INTEGER}
Relationship properties:
ACTED_IN {character: STRING}
The relationships:
(:Movie)-[:BELONGS_TO_GENRE]->(:Genre)
(:Movie)-[:PRODUCED_BY]->(:ProductionCompany)
(:Movie)-[:ORIGIN_COUNTRY]->(:Country)
(:Person)-[:ACTED_IN]->(:Movie)
(:Person)-[:EXECUTIVE_PRODUCER]->(:Movie)
(:Person)-[:DIRECTOR]->(:Movie)
(:Person)-[:PRODUCER]->(:Movie)
```

各类节点信息如下:

节点类型	含义	颜色	数目
Movie	电影		1967
Genre	类型		19
ProductionCompany	制作公司		2611
Country	国家		54
Person	人（演员、导演...）		17540

Table 1: 各类节点含义、颜色、数目

分别查询前 22、前 200 个节点，通过 `yfiles_jupyter_graphs.GraphWidget` 可视化得到电影知识图谱的局部图示如下：

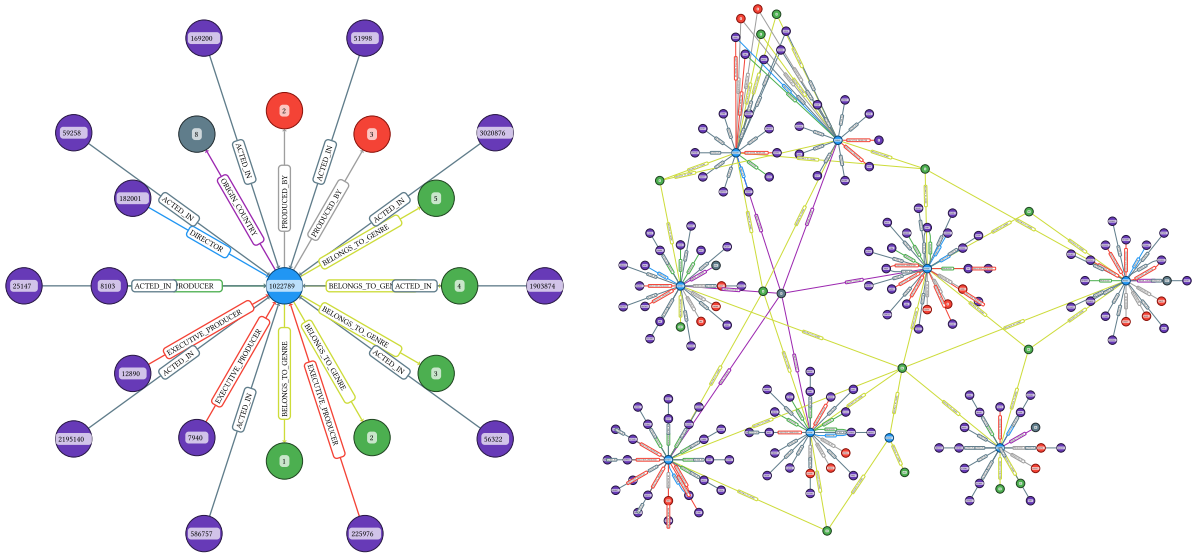

Figure 3: 电影知识图谱局部图示

3 问答系统搭建

在将数据处理存入数据库后，我们基于 LangChain 的 `GraphCypherQAChain`, `ChatZhipuAI`, `GraphCypherQAChain` 模块搭建了一个问答系统，具体代码如下：

```
from langchain_community.chat_models import ChatZhipuAI
from langchain.chains import GraphCypherQAChain
from langchain_community.graphs import Neo4jGraph

llm = ChatZhipuAI(
    temperature=0,
    zhipuai_api_key=os.getenv("ZHIPUAI_API_KEY"),
    model_name="GLM-4-0520",
    # model_name="GLM-4-Flash",
)
graph = Neo4jGraph(
    url="bolt://localhost:7687", username="neo4j",
password=os.getenv("NEO4J_PASSWORD")
)
chain = GraphCypherQAChain.from_llm(
    llm,
```

```
    cypher_prompt=CYPHER_GENERATION_PROMPT,
    qa_prompt=CYPHER_QA_PROMPT,
    graph=graph,
    verbose=True,
)
def get_answer(question, choice=""):
    if choice:
        return chain.invoke(choice)["result"]
    return chain.invoke(question)["result"]
```

其中 CYPHER_GENERATION_PROMPT 指定了 LLM 如何利用数据库的 schema 信息从输入的问题生成 Cypher 语句以对数据库进行查询. 我们在默认 prompt 的基础上添加了几个 question-response 对, 以使得 LLM 对任务的理解更加清晰; 我们还添加了模糊搜索功能, 通过设定相似度阈值, 使得在匹配字符串字段时可以容忍一定的拼写错误. 修改部分如下 (篇幅限制, 完整 prompt 见源码):

```
"""...
Use a similarity threshold of {similarity_threshold} for the levenshteinSimilarity
function when matching string fields.
...
Examples:
question: Who starred in 'The Dark Knight'?
response: MATCH (p:Person)-[:ACTED_IN]->(m:Movie) WHERE
apoc.text.levenshteinSimilarity(m.title, 'The Dark Knight') >= {similarity_threshold}
RETURN p.name;

question: Which movies did Christopher Nolan direct?
response: MATCH (d:Director)-[:DIRECTOR]->(m:Movie) WHERE
apoc.text.levenshteinSimilarity(d.name, 'Christopher Nolan') >=
{similarity_threshold} RETURN m.title;

question: What other films has the director of 'The Dark Knight' directed?
response: MATCH (m:Movie) WHERE apoc.text.levenshteinSimilarity(m.title, 'The Dark
Knight') >= {similarity_threshold} MATCH (m)<-[:DIRECTOR]-(p:Person)-[:DIRECTOR]-
>(otherMovies:Movie) RETURN otherMovies.title
..."""
```

CYPHER_QA_PROMPT 指定了 LLM 如何利用从电影知识图谱中取出的信息来回答输入的问题. 类似地, 我们也添加了例子供 LLM 进行上下文学习 (第二个例子是添加的):

```
"""...
Examples:
Question: Which managers own Neo4j stocks?
Context:[manager:CTL LLC, manager:JANE STREET GROUP LLC]
Helpful Answer: CTL LLC, JANE STREET GROUP LLC owns Neo4j stocks.

Question: Which movies did user born in year 2001 like?
Context:[{{'m.title': 'Avengers 5'}}, {{'m.title': 'The Twilight Saga: New Moon'}},
{{'m.title': 'Rocky'}}]
Helpful Answer: The movies that a user born in 2001 liked include 'Avengers 5', 'The
Twilight Saga: New Moon' and 'Rocky'.
..."""
```

最后, 我们通过 Gradio 部署上面的问答系统.

4 实验效果

我们主要在部分人为撰写的问题上测试该系统. 测试系统时, 我们主要关注两个方面:

- LLM 能否根据数据库 schema 信息正确地从输入问题中提取 Cypher 语句;
- LLM 能否根据从电影知识图谱中查询到的 context 信息正确地回答输入问题.

下面是在撰写的电影知识图谱和用户信息相关问题上的测试结果:

Question	Full Context	Answer	说明
Who starred in 'The Dark Knight'?	[{'p.name': 'Michael Caine'}, {'p.name': 'Maggie Gyllenhaal'}, ...]	The stars of 'The Dark Knight' include Michael Caine, Maggie Gyllenhaal,...	简单查询
Who directed 'The Dark Knight'?	[{'p.name': 'Christopher Nolan'}]	Christopher Nolan directed 'The Dark Knight'	简单查询
Which movies did Christopher Nolan direct?	[{'m.title': 'Interstellar'}, {'m.title': 'Oppenheimer'},...]	Christopher Nolan directed the movies 'Interstellar', 'Oppenheimer',...	简单查询
Which movies did Christopher Noran direct?	[{'m.title': 'Interstellar'}, {'m.title': 'Oppenheimer'},...]	Christopher Nolan directed the movies 'Interstellar', 'Oppenheimer',...	问题中存在拼写错误
Besides 'The Dark Knight', which other movies has Christopher Nolan directed?	[{'m.title': 'Interstellar'}, {'m.title': 'Oppenheimer'},...]	Christopher Nolan directed the movies 'Interstellar', 'Oppenheimer',...	添加筛选条件
In which movies have Michael Caine and Maggie Gyllenhaal co-starred?	[{'m.title': 'The Dark Knight'}]	Michael Caine and Maggie Gyllenhaal have co-starred in 'The Dark Knight'	prompt 中例子未给出
What other films has the director of 'Kingdom of the Planet of the Apes' directed?	[{'otherMovies.title': 'Maze Runner: The Death Cure'}, {'otherMovies.title': 'Maze Runner: The Scorch Trials'}, {'otherMovies.title': 'The Maze Runner'}]	The director of 'Kingdom of the Planet of the Apes' has also directed 'Maze Runner: The Death Cure', 'Maze Runner: The Scorch Trials', and 'The Maze Runner'	prompt 中例子类似问题
What other films has the director of 'Kingdon of the Planett of the Apes' directed?	[{'otherMovies.title': 'Maze Runner: The Death Cure'}, {'otherMovies.title': 'Maze Runner: The Scorch Trials'}, {'otherMovies.title': 'The Maze Runner'}]	The director of 'Kingdom of the Planet of the Apes' has also directed 'Maze Runner: The Death Cure', 'Maze Runner: The Scorch Trials', and 'The Maze Runner'	问题中存在拼写错误
What other films has the director of 'Kingdom of the Apes' directed?	[]	I don't know the answer	问题中电影名缺失过多
What is the revenue of 'Kingdom of the Planet of the Apes'?	[{'m.revenue': 359772773}]	The revenue of 'Kingdom of the Planet of the Apes' is $359,772,773	prompt 中例子未给出

Table 2: 电影知识图谱问答测试结果

从 Table 2, Table 3 测试过程和结果我们观察到:

- 模糊搜索的设置是有效的, 当出现在容忍阈值内的拼写错误, 仍能从数据库中查找到相应的节点, 并且 LLM 在生成回答时, 能够利用其先验知识纠正其中的错误, 但偶尔会将有错误的实体名原样放到输出中.
- 在给定数据库 schema 足够清晰的情况下, LLM 多数时候能够准确地生成 Cypher 语句. 我们最初插入用户节点的方式是将其作为 Person 节点插入, 导致图的结构不够清晰, 因为 crew, cast 中提取出的 Person 节点属性和插入的用户节点属性是两个不同的集合, 不存在包含关系, 这直接导致 LLM 不能生成正确的查询语句.
- 有时 LLM 会在给出 context 情况下拒绝回答, 可能是模型对如何使用 context 中的信息来回答问题没有把握, 我们选择在 prompt 中加入这些拒绝回答的例子, 一定程度上解决了问题.
- 在给出有代表性的例子之后, 对于例子之外的问题, 问答系统也能给出正确答案.

Question	Full Context	Answer	说明
Which directors does User 5 love?	[{'d.name': 'David Gregory'},...]	User 5 loves the directors David Gregory, Olivier Megaton, Katsuhiko Fujii, and Lee Isaac Chung	简单查询
Which movies did user born in year 2001 like?	[{'m.title': 'Avengers 5'}, {'m.title': 'The Twilight Saga: New Moon'},...]	The movies that a user born in 2001 liked include 'Avengers 5', 'The Twilight Saga: New Moon',...	prompt 中例子
Which movies did user born in year 1996 like?	[{'m.title': 'Murder Mystery'}]	The movies that a user born in 1996 liked include 'Murder Mystery'	prompt 中例子类似问题
Which films should person born in year 1972 not watch?	[{'m.title': 'Mobile Suit Gundam SEED FREEDOM'}, {'m.title': 'Stay'}]	I don't know the answer	正确得到了 Cypher 语句并查询到了信息, 但 LLM 没有给出回答
Which user likes the director Christopher Nolan's films?	[{'u.name': 'User 53'}]	User 53 likes the director Christopher Nolan's films.	简单查询
What are the films that people like, give 10 films with their directors?	[{'m.title': 'Poor Things', 'd.name': 'Yorgos Lanthimos'},...]	The films that people enjoy, along with their directors, include 'Poor Things' directed by Yorgos Lanthimos,...	简单查询
Where do user who like the director Christopher Nolan live?	[{'u.location': 'USA'}]	The user who likes the director Christopher Nolan lives in the USA	prompt 中例子
Where do people who like the director Yorgos Lanthimos live?	[{'u.location': 'Australia'}]	People who like the director Yorgos Lanthimos live in Australia	prompt 中例子类似问题
What is the gender distribution of users who love films?	[{'u.gender': 'Male', 'count': 56}, {'u.gender': 'Female', 'count': 65}, {'u.gender': 'Other', 'count': 89}]	The gender distribution of users who love films is 56 males, 65 females, and 89 who identify as other	prompt 中例子未给出
Who are the people that liked Action, Adventure films?	[{'u.name': 'User 5'},...]	The people that liked Action, Adventure films include User 5,...	prompt 中例子未给出

Table 3: 与用户信息相关问答测试结果, 在原电影知识图谱上插入了 `User` 节点和喜恶关系, 实现于 `movies.MoviesKG.add_user_preferences`

5 结论

本项目中, 我们的主要任务是从 TMDB API 获取电影数据, 经过筛选处理后存入 Neo4j 数据库建立知识图谱, 并借助 LangChain 框架和智谱 API, 调整 指导模型生成 Cypher 语句 和 指导模型生成问题答案 的 prompt, 最终, 使用 Gradio 部署用户界面, 实现了一个电影知识图谱问答系统. 当然, 由于时间限制, 这个系统还有一些无法忽视的瑕疵, 我们将在未来着手改进. 目前我们认为的不足之处主要有:

- 模糊检索准则单一, 缺乏交互过程. 目前我们的系统仅支持以 Levenshtein 相似度阈值为准则进行模糊匹配, 且模糊匹配结果的确认缺乏交互. 这会导致一些明显的问题, 比如对于某些以序号命名的系列电影, 系列内电影编辑距离很小 (多数情况仅为 1), 以上面的准则进行模糊搜索将返回所有系列电影, 某些情况下可能并不是我们想要的. 另外, 这个模糊搜索是在 Cypher 语句中实现的, LLM 不能获得数据库查询的中间结果, 而仅能根据自身先验知识纠正拼写错误. 理想的过程是系统应该能对用户输入问题中的实体 (如电影名) 给出多个候选项, 用户进行选择后再继续完成任务.
- 系统评估不够全面. 目前我们仅在撰写的少数问题上评估了我们的系统, 难免存在认知方面的遗漏, 因而评测结果可能并不能客观地反映系统的真实能力.
- 未充分利用知识图谱的图结构本质挖掘更多信息. 目前我们的问答系统主要基于 LLM 驱动的数据库查询, 没有利用到图的度、连通性等结构信息. 探索如何结合图的结构信息, 将使系统具备一定的推荐能力.

未来, 我们将会致力于从以上几个方面改进系统, 以期进一步提升电影知识图谱问答系统的准确性、交互性和智能性，为用户提供更优质的服务。

参考资源

[1] V. Daga, “How to Build a Movie Recommendation System Powered by Knowledge Graph FalkorDB.” 2024.

[2] T. Bratanic, “Enhancing the Accuracy of RAG Applications With Knowledge Graphs.” 2024.

[3] A. Brams, “Movie Recommendations powered by Knowledge Graphs and Neo4j.” 2024.

[4] Neo4j, “Neo4j documentation.” 2024.

[5] LangChain, “LangChain documentation.” 2024.

[6] 智谱 AI, “智谱 AI 开放平台 - 开发者中心 - 文档.” 2024.

小组成员分工

组内成员	负责内容
陈抚民	主要负责数据收集、处理及报告撰写
涂伟豪	主要负责数据库和 QA 系统的搭建和 prompt 改写、测试
李龙昊	主要负责展示和报告撰写

Table 4: 小组成员分工